  ****

**An Intellectual Model for Detecting Respratory Disease Using Sound Pattern**

**A PROJECT REPORT**

***Submitted by***

**Mohammed Ameen P  
 Mohamed Shameer M.I  
 Megash S**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PANIMALAR ENGINEERING COLLEGE, CHENNAI-600123.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**APRIL 2021**

**BONAFIDE CERTIFICATE**

Certified that this project report “**SOUND CLASSIFICATION FOR RESPIRATORY DISEASES USING MACHINE LEARNING TECHNIQUE**” is the bonafide work of “**DAKSHNA MOORTHY P (211417104046), HARIKRISHNA M (211417104076), JUDE MATHEW (211417104102)**” who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

Certified that the above candidate(s) was/ were examined in the Anna University

Project Viva-Voce Examination held on...........................

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We express our deep gratitude to our respected Secretary and Correspondent **Dr.P.CHINNADURAI, M.A., Ph.D.** for his kind words and enthusiastic motivation, which inspired us a lot in completing this project.

We would like to express our heartfelt and sincere thanks to our Directors Tmt.C.VIJAYARAJESWARI, **Dr.C.SAKTHIKUMAR, M.E., Ph.D.,** and **Tmt. SARANYASREE SAKTHIKUMAR B.E., M.B.A.,** for providing us with the necessary facilities for completion of this project.

We also express our gratitude to our Principal **Dr.K.Mani, M.E., Ph.D.** for his timely concern and encouragement provided to us throughout the course.

We thank the HOD of CSE Department, **Dr. S.MURUGAVALLI, M.E.,Ph.D.,** for the support extended throughout the project.

We would like to thank my **Project Guide Mr.N.Sathish, M.E.** and all the faculty members of the Department of CSE for their advice and suggestions for the successful completion of the project.

DAKSHNA MOORTHY P

HARIKRISHNA M

JUDE MATHEW

**ABSTRACT**

Respiratory sounds are one of the important signs of lung health and respiratory disorders. These respiratory sounds can acquire using digital stethoscopes and other recording devices. This advanced information opens up the chance of utilizing artificial intelligence (AI) to naturally analyze respiratory scatters like asthma, pneumonia and bronchiolitis, to give some examples. A very high number of people lose their lives to different respiratory diseases every day. Respiratory Sound Analysis has been a key tool to accurately detect these types of diseases. Earlier manual detection of respiratory sounds was used but it is not feasible to detect various lung diseases due to various reasons like audio quality and perceptions of different doctors. Modern computer aided analysis helps to give much better results in identifying the diseases from the sound i.e. identification of wheezes and crackles and thus better treatment can be given to patients. These respiratory sound diseases include Asthma, Bronchitis, Pneumonia, COPD and URTI. The prediction with decision trees gives an accuracy rate of 90 percent. This research will be very helpful for the healthcare professionals such as doctors for the easy and accurate diagnosis of respiratory diseases. This study will be a major contribution in the area of the respiratory disease classification by using lungs sounds. In this project a web application is created with ease of access for the user. The user will login to this application and gives patient’s data as input. The machine learning model will predict the disease patient suffering from. If the patient does not have disease it will display as healthy.

**LIST OF TABLES**

**CHAPTER NO. TITLE PAGE NO.**

**3. SYSTEM ANALYSIS**

3.1 Minimum Hardware Requirements 18

3.2 Minimum Software Requirement 18

**4. SYSTEM DESIGN**

4.1. Data Dictionary of User 27

**6. SYSTEM TESTING**

6.1. Test Case for User 43

**LIST OF FIGURE**

**CHAPTER NO. TITLE PAGE NO.**

**4. SYSTEM DESIGN**

4.1. ER-Diagram 26

4.2. Data Flow Diagram of 28

4.4. Use Case Diagram 32

4.5. Class Diagram 33

4.6 Sequence Diagram 34

**5. SYSTEM ARCHITECTURE**

5.1. System Architecture 36

**TABLE OF CONTENTS**

**CHAPTER NO. TITLE PAGE NO.**

**ABSTARCT** iv

**1. INTRODUCTION** 1

1.1 Overview 2

**2. LITERATURE SURVEY** 10

**3. SYSTEM ANALYSIS** 15

3.1 Existing System 16

3.2 Proposed system 16

3.3 Requirement Analysis and Specification 17

3.3.1 Input Requirements 17

3.3.2 Output Requirements 17

3.3.3 Functional Requirements 17

3.4 Feasibility Study 17

3.4.1 Technical Feasibility 18

3.4.2 Economic Feasibility 18

3.5 Hardware Requirements 18

3.6 Software Requirements 18

3.7 Software Specification 19

**CHAPTER NO. TITLE PAGE NO.**

**4. SYSTEM DESIGN** 25

4.1. ER diagram 26

4.2 Data dictionary 26

4.3 Data Flow Diagram 27

4.4 UML Diagrams 28

**5. SYSTEM ARCHITECTURE** 35

5.1 Architecture Overview 36

5.2 Module Design Specification 36

5.2.1 Modules Explanation 37

5.3 Program Design Language 38

**6. TESTING** 40

6.1 System Testing 41

6.2 Web Application Testing 42

6.3 Unit Testing 43

6.4 Test Cases & Reports / Performance Analysis 43

**7. CONCLUSION** 45

7.1 Conclusion 46

7.2 Future Enhancements 46

**APPENDICES** 47

A.1 Sample Screens 47

A.2 Sample Code 50

**REFERENCES** 83

**CHAPTER 1**

**INTRODUCTION**

**1.1 OVERVIEW**

The classification and identification of breathing diseases is a tedious task. The sound that is produced when a person breathes is directly associated with the movement of air, variations in the lung tissue and the position of the secretions inside the lung. A wheezing sound is an example for a person with obstructive disease like asthma or chronic obstructive pulmonary disease (COPD). One of the major causes of mortality and morbidity worldwide is respiratory diseases. It developed the third prominent cause of death in 2020. Each year greater than 3 million of people go down from chronic obstructive pulmonary diseases (COPDS) which is roughly 6% of all fatalities in the world. Around 251 million cases of chronic obstructive pulmonary disease (COPD) were reported globally in 2016. By 2030 COPD will be one of the leading causes of death worldwide. Asthma is also related to COPD, but the definition is different. This disease also results in social and economic burden that is both substantial and increasing.

The important treatment outcomes of COPD are symptoms, acute exacerbations and limitations of airflow .Interestingly, the Sounds from the lungs conveys significant information associated with respiratory diseases and it helps to assess the patients with pulmonary or respiratory disorders. Sounds released from a person’s breath are directly related to changes in lung tissue, position of secretion within the lungs and air movement. For instance, wheezing sound is a common indication that the patients have diseases like obstructive airway disease (asthma and chronic obstructive pulmonary disease). The healthcare professionals use traditional auscultation methods to detect the disorders of lungs, but this method has many limitations for instance there are chances of misdiagnosis if the physician is not trained well. In addition, lungs are non-stationary, this leads to difficulty in recognition, analysis and distinction. Pulmonary disease affects social, economic and health of people’s lives significantly. Because of these reasons lots of research are going on for early diagnosis and intervention of respiratory disease. In this perspective, characteristics of lung sounds provide valuable indications for the diagnosis and detection of respiratory abnormalities and infections.

However, there are many limitations in the application of stethoscope in research studies because of the variability in inter-observer and subjectivity in lung sounds interpretation. Diagnosis of the diseases from lung sounds needs professional training and experts. This is undoubtedly costly and inconvenient. In this context a technique that can automatically and accurately classify the sounds of lungs into many groups is very meaningful. It helps to detect potential threats at a very early stage. So, the most important purpose of the research is to develop an automated system to predict and diagnose respiratory diseases using lung sounds.

The most important objective of the research is to detect and categorize the lung noise digital signal with the help of signal learning processing methods. More specifically the study will be comparing the performance of convolutional neural network architecture and machine learning algorithms and to create a model based on the best performing algorithm. Ideally, this technique will improve detection of sounds and categorization of accuracy and robustness when encountered with different modes of sound and additional components while gaining the lung vibration wave. This research will be very helpful for the healthcare professionals such as doctors for the easy and accurate diagnosis of respiratory diseases. This study will be a major contribution in the area of the respiratory disease classification by using lungs sounds. It serves as a stepping- stone for future research in classification of lung sounds using convolution neural networks. In addition, it helps the policy makers and researchers to make and amend the decisions in lung diseases. We use the python as a background for our development of the system as it gives more functionality for data analysis.

**TYPES OF DISEASES**

**ASTHMA**

Asthma is a condition in which your airways narrow and swell and may produce extra mucus. This can make breathing difficult and trigger coughing, a whistling sound (wheezing) when you breathe out and shortness of breath. For some people, asthma is a minor nuisance. For others, it can be a major problem that interferes with daily activities and may lead to a life-threatening asthma attack. Asthma can't be cured, but its symptoms can be controlled. Because asthma often changes over time, it's important that you work with your doctor to track your signs and symptoms and adjust your treatment as needed.

**BRONCHITIS**

The most common cause of chronic bronchitis is cigarette smoking**.** Air pollution and dust or toxic gases in the environment or workplace also can contribute to the condition. Chronic bronchitis results from Trusted Source repeated irritation and damage to the lung and airway tissues. The most common cause is smoking, but not everyone with bronchitis is a smoker. Bronchitis is an inflammation of the lining of your bronchial tubes, which carry air to and from your lungs. People who have bronchitis often cough up thickened mucus, which can be discolored. Bronchitis may be either acute or chronic.

**PNEUMONIA**

Pneumonia is a lung infection that can range from mild to so severe that you have to go to the hospital. It happens when an infection causes the air sacs in your lungs (your doctor will call them alveoli) to fill with fluid or pus. That can make it hard for you to breathe in enough oxygen to reach your bloodstream. Lifestyle habits, like smoking cigarettes and drinking too much alcohol, can also raise your chances of getting pneumonia.

**CHRONIC OBSTRUCTIVE PULMONARY DISEASE**

Chronic obstructive pulmonary disease (COPD) is a type of [obstructive lung disease](https://en.wikipedia.org/wiki/Obstructive_lung_disease) characterized by long-term breathing problems and poor airflow. The main symptoms include [shortness of breath](https://en.wikipedia.org/wiki/Shortness_of_breath) and [cough](https://en.wikipedia.org/wiki/Cough) with [sputum](https://en.wikipedia.org/wiki/Sputum) production. COPD is a [progressive disease](https://en.wikipedia.org/wiki/Progressive_disease), meaning it typically worsens over time. Eventually, [everyday activities](https://en.wikipedia.org/wiki/Activities_of_daily_living) such as walking or dressing become difficult. [Chronic bronchitis](https://en.wikipedia.org/wiki/Bronchitis#Chronic_bronchitis) and [emphysema](https://en.wikipedia.org/wiki/Pneumatosis#Lungs) are older terms used for different types of COPD. The term "chronic bronchitis" is still used to define a productive cough that is present for at least three months each year for two years. Those with such a cough are at a greater risk of developing COPD.

**LRTI**

Lower respiratory tract infection (LRTI) is a broad terminology which includes acute bronchitis, pneumonia, acute exacerbations of chronic obstructive pulmonary disease/chronic bronchitis (AECB), and acute exacerbation of bronchiectasis.

Infections are caused by tiny organisms known as bacteria or viruses, which are usually the most common cause. They are carried in tiny droplets and passed between people by coughing, sneezing and at times by indirect contact with surfaces. People who have lower respiratory tract infections will experience coughing as the primary symptom. People with upper respiratory tract infections will feel the symptoms mainly above the neck, such as sneezing, headaches, and sore throats.

**STETHOSCOPE**

An acoustic device used to detect the breath sounds of a patient is the stethoscope, a diaphragm in stethoscope can detect normal breath sounds and abnormal breath sounds without increasing lower frequency masking sounds. A hollow tube fitted to a chest piece including a wider diaphragm is in the currently available acoustic stethoscope. The high frequency sounds are transmitted through diaphragm and the low wavelength sounds are transmitted through smaller hollow bell. This kind of stethoscope will convey sound proportional to frequency made from the heart sounds. A digital stethoscope tries to better on some of the functionalities of the traditional acoustic stethoscope. The sounds are converted into digital analogue by using piezoelectric sensor or a microphone which capture and transform the sound to electrical. The acquired signal is then passed through a band pass filter to amplify and process the signal and to remove noise that is corrupting the signal. Another round of denoising is finished with a computerized channel, to extract the signals of interest from the frequency band. Some propelled curio evacuation procedures are added here. The heart sounds acquired in this process are standardized to a specific scale and are fragmented into cycles. This causes us in identifying the sound segments obviously.

## 

## **RESPIRATORY SOUNDS**

Identification of normal and abnormal respiratory sounds such as crackles, wheezes is very essential for accurate diagnosis of the diseases. These sounds include a lots of information about the pathologies and physiologies of lung structure and any obstruction in airways can be identified from the sounds. Around the beginning of the 19th century, doctors diagnosed their patients by keeping their ears to the thorax and chest to hear the noises with in and this method was called immediate auscultation. Various studies were done, and research was made to test human ears' capacity to identify crackles. The research consisted of crackles simulated to superimpose as real respiratory/breathing sound.

The most important detection errors were identified from these research like intensity of crackles, different types of crackles, different wavelengths and so on. From these studies we could conclude that traditional auscultation should not be considered as individual reference for validating respiratory sounds.

The following two sounds explore the difference between lung and breath sounds.

**Lung Sounds**

Air flow of the chest causes the respiratory sounds. Pulmonary deficiencies result in the changes of the lung sounds.

**Breath sounds**

Airflow through the trachea-bronchial tree. According to Hadjileontiadis and Moussavi (2018), breath sounds are crackling, plashing, wheezing and bubbling sounds coming from the chest.

Respiratory sounds are classed as normal and adventitious or abnormal sounds.

**Normal respiratory sounds**

The normal breath sounds are categorized as vesicular, bronchial, or Broncho vesicular. Based on anatomical features of the location where you are auscultating these sounds have different acoustic characteristics. These sounds are produced from healthy lungs.

**Vesicular sounds**

It is the normal noise of the breath heard over the chest wall and concurrent with the airways air flow. It is originated from alveoli. These sounds can hear only in the starting of expiration and these are easy to hear during inspiration. Zimmerman and Williams (2020) acknowledge that vesicular sounds are low pitched, soft, mostly inspiratory and appreciated well at the posterior lung bases

**Bronchial sounds**

It is also called tubular sounds, and these are arising from tracheobronchial tree. Sounds like rough, noisy and high-pitched respiratory sounds are received over the trachea.

**Broncho vesicular sounds**

During inspiration and expiration time this sound can be hear and it have a mid-range intensity and pitch. Commonly these sounds are heard over the upper 3rd of the anterior chest.

**Abnormal sounds**

Abnormal heart sounds are heard in addition to the normal respiratory sounds that are mentioned in the above section 2.5.1. it is also called adventitious sounds that is, sound super imposed on normal breath sounds. These sounds can categorized on the basis of underlying condition and therefore it is very useful in aiding diagnosis. Based on the duration of sounds abnormal sounds can be grouped into continuous and discontinuous sounds. Crackle, Rhonchi and wheeze are the main abnormal sounds that are usually heard. Rhonchi and wheeze comes under the continuous sounds because they are inseparable sounds and are not interrupted while crackles are discontinuous sounds as it is calculated by assessor as decreased acoustic event it’s like dropping a marble on a floor.

**Rhonchi**

Rhonchi sounds are low-pitched sounds having rapidly damping periodic waveform. It has more than 100 ms duration and less than 300 Hz frequency. These sounds are related to creation of breach of fluid film and airway collapsibility. These sounds can be clearly head during coughing and clinical practice evident that secretions in large airways contribute a significant role in producing Rhonchi sounds. These sounds can be measured as a marker of constriction of airway lumen and mucosal thickening, oedema or bronchospasm. Zimmerman and Williams (2020) highlight that Rhonchi sounds are usually arise during exhalation or both inhalation and exhalation time, but it do not arise in inhalation alone.

**Crackle**

The nature of crackle sound is sudden bursts and explosive. Less than 20 ms is the duration of crackle sounds and has 200-2000 Hz spectrum of frequency range. These are produced during the closing in the expiration or during the opening of abnormally closed airway at the time of inspiration. Each of the immediate closing or opening of an airway represents single crackle. Forgac’s theory states that a gas pressure is developed across the airway during inspiration which is then collapse during expiration.

Based on the duration, scheduling in the respiratory cycles, pitch, loudness and relation to altering the status of body and coughing crackles are classed as fine and coarse crackles. Inside the small airways fine crackles are generated and having shorter duration 5 ms, while medium crackle are supplied by bubbling of air via secretion in small bronchi. Coarse crackles are delivered from bronchiectasis segments or from large bronchi and it has longer duration 15 ms. The combination of fine and coarse crackle is called biphasic crackle. With the help of time expanded waveform analysis crackles can be differentiated.

**Wheezes**

These are melodic sounds that are generated by movement of air through bronchioles or constricted small airways. At the end of inspiratory phase or early expiratory phase wheeze are typically heard. These can be regarded as a marker for detection of obstructive disease such as COPD and asthma also patients having sickle cell experience an acute pain crisis.it is produced when gradual closure during expiration and reopening at the time of inspiration. 100-250 ms is the duration of wheeze and these are characterized as sinusoidal oscillations with harmonic distortion. Between 2- 7 tracheobronchial tree wheeze is more likely occurs.

**CHAPTER 2**

**LITERATURE SURVEY**

**Emmanuel Andres, Amir Hajjam ‘Advances and Perspectives in the Field of Auscultation, with a Special Focus on the Contribution of New Intelligent Communicating Stethoscope’**

The stethoscope and the semantic of auscultatory findings were invented more than 200 years ago by Dr. Laennec and over the years very few changes have been made to both the stethoscope itself and the way in which it is used. . However, the ability to differentiate between normal and abnormal sounds or noises (vesicular sounds, wheezes, crackles, etc.) remains essential in clinical practice for correct diagnosis and management. Over the past two decades, much of the progress made in this area has resulted primarily from improvements made to the stethoscope itself. More recently, we have seen advances in the techniques used to process auscultatory signals as well as in the analysis and clarification of the resulting sounds. The availability of novel representations of the sounds, with phono- and spectrograms, also opens interesting perspectives in the context of diagnostic aids, but also in education and pedagogy. It aims to review recent technological advances, evaluate promising innovations and perspectives in the field of auscultation, with a special focus on the development of new intelligent communicating stethoscope systems in clinical practice, and in the context of teaching and telemedicine.

**B. M. Rocha, D. Filos, L. Mendes, I. Vogiatzis, E. Perantoni, E. Kaimakamis, P. Natsiavas, A. Oliveira, C. Jácome, A. Marques, R. P. Paiva, I. Chouvarda, P. Carvalho, N. MaglaverasΑ. ‘Respiratory Sound Database for the Development of Automated Classification’**

The automatic analysis of respiratory sounds has been a field of great research interest during the last decades. Automated classification of respiratory sounds has the potential to detect abnormalities in the early stages of respiratory dysfunction and thus enhance the effectiveness of decision making. However, the existence of a publically available large database, in which new algorithms can be implemented, evaluated, and compared, is still lacking and is vital for further developments in the field. In the context of the International Conference on Biomedical and Health Informatics (ICBHI), the first scientific challenge was organized with the main goal of developing algorithms able to characterize respiratory sound recordings derived from clinical and non-clinical environments. The database was created by two research teams in Portugal and in Greece, and it includes 920 recordings acquired from 126 subjects. A total of 6898 respiration cycles were recorded. The cycles were annotated by respiratory experts as including crackles, wheezes, a combination of them, or no adventitious respiratory sounds. The recordings were collected using heterogeneous equipment and their duration ranged from 10 to 90 s. The chest locations from which the recordings were acquired was also provided. Noise levels in some respiration cycles were high, which simulated real life conditions and made the classification process more challenging.

**Renars Xaviero Adhi Pramono, Stuart bowyer, Esther Rodriguez-Villegas ‘Automatic Adventitious Respiratory Sound Analysis: A Systematic Review’**

Background Automatic detection or classification of adventitious sounds is useful to assist physicians in diagnosing or monitoring diseases such as asthma, Chronic Obstructive Pulmonary Disease (COPD), and pneumonia. While computerized respiratory sound analysis, specifically for the detection or classification of adventitious sounds, has recently been the focus of an increasing number of studies, a standardised approach and comparison has not been well established. Objective To provide a review of existing algorithms for the detection or classification of adventitious respiratory sounds. This systematic review provides a complete summary of methods used in the literature to give a baseline for future works. Search terms included adventitious sound detection, adventitious sound classification, abnormal respiratory sound detection, abnormal respiratory sound classification, wheeze detection, wheeze classification, crackle detection, crackle classification, rhonchi detection, rhonchi classification, stridor detection, stridor classification, pleural rub detection, pleural rub classification, squawk detection, and squawk classification. Detection or classification methods used varied from empirically determined thresholds to more complex machine learning techniques.

**Singh, A., Thakur, N. and Sharma, A. ‘A Review of Supervised Machine Learning Algorithms’.**

Supervised machine learning is the construction of algorithms that are able to produce general patterns and hypotheses by using externally supplied instances to predict the fate of future instances. Supervised machine learning classification algorithms aim at categorizing data from prior information. Classification is carried out very frequently in data science problems. Various successful techniques have been proposed to solve such problems viz. Rule-based techniques, Logic-based techniques, Instance-based techniques, stochastic techniques. This paper discusses the efficacy of supervised machine learning algorithms in terms of the accuracy, speed of learning, complexity and risk of overfitting measures. The main objective of this paper is to provide a general comparison with state of art machine learning algorithms.

**Georgios Ntritsos, Jacob Franek, Lazaros Belbasis, Maria A Christou, Georgios Markozannes, Pablo Altman, Robert Fogel, Tobias Sayre,Evangelia E Ntzani, Evangelos Evangelou. ‘Gender-specific estimates of COPD prevalence: a systematic review and meta-analysis’**

COPD has been perceived as being a disease of older men. However, >7 million women are estimated to live with COPD in the USA alone. Despite a growing body of literature suggesting an increasing burden of COPD in women, the evidence is limited. To assess and synthesize the available evidence among population-based epidemiologic studies and calculate the global prevalence of COPD in men and women. A systematic review and meta-analysis reporting gender-specific prevalence of COPD was undertaken. Gender-specific prevalence estimates were abstracted from relevant studies. Associated patient characteristics as well as custom variables pertaining to the diagnostic method and other important epidemiologic covariates were also collected. A Bayesian random-effects meta-analysis was performed investigating gender-specific prevalence of COPD stratified by age, geography, calendar time, study setting, diagnostic method, and disease severity. Among 194 eligible studies, summary prevalence was 9.23% (95% credible interval [CrI]: 8.16%–10.36%) in men and 6.16% (95% CrI: 5.41%–6.95%) in women.We conducted the largest ever systematic review and meta-analysis of global prevalence of COPD and the first large gender-specific review. These results will increase awareness of COPD as a critical woman’s health issue.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

The existing system mainly focuses on finding the respiratory sounds that have the potential to detect abnormalities in the early stages of respiratory dysfunction and thus enhance the effectiveness of decision making. However, the existence of a publically available large database, in which new algorithms can be implemented, evaluated, and compared, is still lacking and is vital for further developments in the field. The recordings were collected using heterogeneous equipment and their duration ranged from 10 to 90s. The chest locations from which the recordings were acquired was also provided.

**3.2 PROPOSED SYSTEM**

The proposed method uses the identification of the respiratory disease with the help of the datasets that consists of the people with breathing problems. It identifies and tells us the exact respiratory problem that occurs for the individuals. The prediction is exact and all the algorithms used are highly efficient in finding the problem with the help of the spectrography frequencies. This helps in knowing the problem prior to the last stage.

**3.2.1 Advantages**

1. Prediction is easy.
2. Noise is eliminated.
3. Used to predict more diseases than the existing system.
4. Cheaper compared to the existing system.

**3.3 REQUIREMENT ANALYSIS AND SPECIFICATION**

The requirement engineering process of feasibility study, requirements elicitation and analysis, requirement specification, requirements validation and requirement management. Requirement elicitation and analysis is an iterative process that can be represented as a spiral of activities, namely requirements discovery, requirements classification and organization, requirement negotiation and requirements documentation.

**3.3.1 INPUT REQUIREMENT**

The input requirement at the base requires data from users such as the patient's age, gender, BMI (Body Mass Index) and the audio format file that helps the ML model to predict the type of disease.

**3.3.2 OUTPUT REQUIREMENT**

The output requirement depends on the dataset given by the patient and it shows whether the patient is healthy or not as well as what type of respiratory disease the patient suffers.

**3.4 FEASIBILITY STUDY**

A feasibility study is carried out to select the best system that meets performance requirements. The main aim of the feasibility study activity is to determine that it would be financially and technically feasible to develop the product.

**3.4.1 TECHNICAL FEASIBILITY**

This is concerned with specifying the software will successfully satisfy the user requirement. Open source and business-friendly and it is truly cross platform, easily deployed and highly extensible.

**3.4.2 ECONOMIC FEASIBILITY**

Economic analysis is the most frequently used technique for evaluating the effectiveness of a proposed system. The enhancement of the existing system doesn’t incur any kind of increase in the expenses. Programming Language for Web-App development is open source and readily available for all users. Since the project is implemented in Anaconda (Jupyter Notebook) it is cost efficient.

**3.5 HARDWARE REQUIREMENTS**

|  |  |
| --- | --- |
| HARD DISK | >90GB |
| PROCESSOR | >Core i3 2.4GHz |
| SYSTEM TYPE | 32bit / 64 bit |
| RAM | >2GB |

**3.6 SOFTWARE REQUIREMENTS**

|  |  |
| --- | --- |
| OPERATING SYSTEM | WINDOWS 7/8/8.1/10 |
| INTEGRATED DEVELOPMENT KIT | ANACONDA V2019 |
| PROGRAMMING LANGUAGE  (BACKEND) | PYTHON V 3.6 |
| PROGRAMMING LANGUAGE  (FRONTEND) | HTML, CSS |
| DATABASE | PYTHON SQLITE 3 |
| APPLICATION PROGRAMMING INTERFACE (API) | SCIKIT LEARN, FLASK |

**3.7 SOFTWARE SPECIFICATION**

**3.7.1 Anaconda**

Anaconda is a [distribution](https://en.wikipedia.org/wiki/Software_distribution) of the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) and [R](https://en.wikipedia.org/wiki/R_(programming_language)) programming languages for [scientific computing](https://en.wikipedia.org/wiki/Scientific_computing) ([data science](https://en.wikipedia.org/wiki/Data_science), [machine learning](https://en.wikipedia.org/wiki/Machine_learning) applications, large-scale data processing, [predictive analytics](https://en.wikipedia.org/wiki/Predictive_analytics), etc.), that aims to simplify [package management](https://en.wikipedia.org/wiki/Package_management) and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS. It is developed and maintained by Anaconda, which was founded by Peter Wang and [Travis Oliphant](https://en.wikipedia.org/wiki/Travis_Oliphant) in 2012.As an Anaconda, inc product, it is also known as Anaconda Distribution or Anaconda Individual Edition, while other products from the company are Anaconda Team Edition and Anaconda Enterprise Edition, both of which are not free. Package versions in Anaconda are managed by the [package management system](https://en.wikipedia.org/wiki/Package_manager) [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)). This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, python, the packages they depend on, and a small number of other packages. Anaconda distribution comes with over 250 packages automatically installed, and over 7,500 additional open-source packages can be installed from [PyPI](https://en.wikipedia.org/wiki/Python_Package_Index) as well as the [conda](https://en.wikipedia.org/wiki/Conda_(package_manager)) package and virtual environment manager. It also includes a GUI, Anaconda Navigator, as a graphical alternative to the command line interface (CLI).

**3.7.2 Python**

Python is an [interpreter](https://en.wikipedia.org/wiki/Interpreted_language), [high-level](https://en.wikipedia.org/wiki/High-level_programming_language) and [general-purpose programming language](https://en.wikipedia.org/wiki/General-purpose_programming_language). Python's design philosophy emphasizes [code readability](https://en.wikipedia.org/wiki/Code_readability) with its notable use of [significant indentation](https://en.wikipedia.org/wiki/Off-side_rule). Its [language constructs](https://en.wikipedia.org/wiki/Language_construct) and [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) approach aim to help [programmers](https://en.wikipedia.org/wiki/Programmers) write clear, logical code for small and large-scale projects. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library. Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP. Python is Interactive − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs. Python is Object-Oriented − Python supports Object-Oriented style or technique of programming that encapsulates code within objects. Python is a Beginner's Language − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**3.7.3 HTML**

Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets (CSS) and scripting languages such as JavaScript. Web browsers receive HTML documents from a web server or from local storage and render the documents into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects such as interactive forms may be embedded into the rendered page. HTML provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. HTML elements are delineated by tags, written using angle brackets. Tags such as <img /> and <input /> directly introduce content into the page. Other tags such as <p> surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page. HTML can embed programs written in a scripting language such as JavaScript, which affects the behavior and content of web pages. Inclusion of CSS defines the look and layout of content. The World Wide Web Consortium (W3C), former maintainer of the HTML and current maintainer of the CSS standards, has encouraged the use of CSS over explicit presentational HTML since 1997. HTML markup consists of several key components, including those called tags (and their attributes), character-based data types, character references and entity references. HTML tags most commonly come in pairs like <h1> and </h1>, although some represent empty elements and so are unpaired, for example <img>. The first tag in such a pair is the start tag, and the second is the end tag (they are also called opening tags and closing tags). HTML documents imply a structure of nested HTML elements. These are indicated in the document by HTML tags, enclosed in angle brackets thus: In the simple, general case, the extent of an element is indicated by a pair of tags: a "start tag" <p> and "end tag"</p>. The text content of the element, if any, is placed between these tags.

**3.7.4 CSS**

Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript. CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts. This separation can improve content accessibility, provide more flexibility and control in the specification of presentation characteristics, enable multiple web pages to share formatting by specifying the relevant CSS in a separate .css file, and reduce complexity and repetition in the structural content.

Separation of formatting and content also makes it possible to present the same markup page in different styles for different rendering methods. CSS also has rules for alternate formatting if the content is accessed on a mobile device. The name cascading comes from the specified priority scheme to determine which style rule applies if more than one rule matches a particular element. This cascading priority scheme is predictable.

**3.7.5 FLASK**

Flask is a web framework used for developing web applications. It has the ability compile with complex models and get started fast. Only suggestions are given by theFlask, but no dependencies are enforced to application. The availability of many extensions makes the addition of new functionalities simple.

In our web application the users are area sked first to register and create login credentials for them. This done so that the details uploaded by the user is not accessible by other clients and to help for future references.

Our system uses the best performing algorithm from machine learning Decision tree for the analysis of client data. POST method is used to take the data from the user. The webapp design is made in such a way that anyone can easily understand the functionalities. The users are given two options to check the diseases. In the first option the user can input the crackles and wheezes number along with age, gender and body mass index to get the result. The second option is to upload the sound file where the user can get the result from the audio file.

**3.7.6 PANDAS**

Pandas is a [software library](https://en.wikipedia.org/wiki/Software_library) written for the [Python programming language](https://en.wikipedia.org/wiki/Python_(programming_language)) for data manipulation and analysis. In particular, it offers [data structures](https://en.wikipedia.org/wiki/Data_structure) and operations for manipulating numerical tables and [time series](https://en.wikipedia.org/wiki/Time_series). It is [free software](https://en.wikipedia.org/wiki/Free_software) released under the [three-clause BSD license](https://en.wikipedia.org/wiki/3-clause_BSD_license). The name is derived from the term "[panel data](https://en.wikipedia.org/wiki/Panel_data)", an [econometrics](https://en.wikipedia.org/wiki/Econometrics) term for [data sets](https://en.wikipedia.org/wiki/Data_set) that include observations over multiple time periods for the same individuals. Its name is a play on the phrase "Python data analysis" itself. [Wes McKinney](https://en.wikipedia.org/wiki/Wes_McKinney) started building what would become pandas at [AQR Capital](https://en.wikipedia.org/wiki/AQR_Capital) while he was a researcher there from 2007 to 2010.

Pandas is mainly used for [data analysis](https://en.wikipedia.org/wiki/Data_analysis). Pandas allows importing data from various file formats such as [comma-separated values](https://en.wikipedia.org/wiki/Comma-separated_values), [JSON](https://en.wikipedia.org/wiki/JSON), [SQL](https://en.wikipedia.org/wiki/SQL), [Microsoft Excel](https://en.wikipedia.org/wiki/Microsoft_Excel). Pandas allows various data manipulation operations such as merging, reshaping, selecting, as well as [data cleaning](https://en.wikipedia.org/wiki/Data_cleaning), and [data wrangling](https://en.wikipedia.org/wiki/Data_wrangling) features.

**3.7.7 SCIKIT-LEARN**

Scikit-learn is a [free software](https://en.wikipedia.org/wiki/Free_software) [machine learning](https://en.wikipedia.org/wiki/Machine_learning) [library](https://en.wikipedia.org/wiki/Library_(computing)) for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) [programming language](https://en.wikipedia.org/wiki/Programming_language). It features various [classification](https://en.wikipedia.org/wiki/Statistical_classification), [regression](https://en.wikipedia.org/wiki/Regression_analysis) and [clustering](https://en.wikipedia.org/wiki/Cluster_analysis) algorithms including [support vector machines](https://en.wikipedia.org/wiki/Support_vector_machine), [random forests](https://en.wikipedia.org/wiki/Random_forests), [gradient boosting](https://en.wikipedia.org/wiki/Gradient_boosting), [*k*-means](https://en.wikipedia.org/wiki/K-means_clustering) and [DBSCAN](https://en.wikipedia.org/wiki/DBSCAN), and is designed to interoperate with the Python numerical and scientific libraries [NumPy](https://en.wikipedia.org/wiki/NumPy) and [SciPy](https://en.wikipedia.org/wiki/SciPy).

Scikit-learn is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modeling including classification, regression, and clustering and dimensionality reduction via a consistence interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

**3.7.8 WERKZEUG**

Werkzeug is a comprehensive WSGI web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.

It includes:

* An interactive debugger that allows inspecting stack traces and source code in the browser with an interactive interpreter for any frame in the stack.
* A full-featured request object with objects to interact with headers, query args, form data, files, and cookies.
* A response object that can wrap other WSGI applications and handle streaming data.
* A routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs.

HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more.

A threaded WSGI server for use while developing applications locally.

A test client for simulating HTTP requests during testing without requiring running a server.

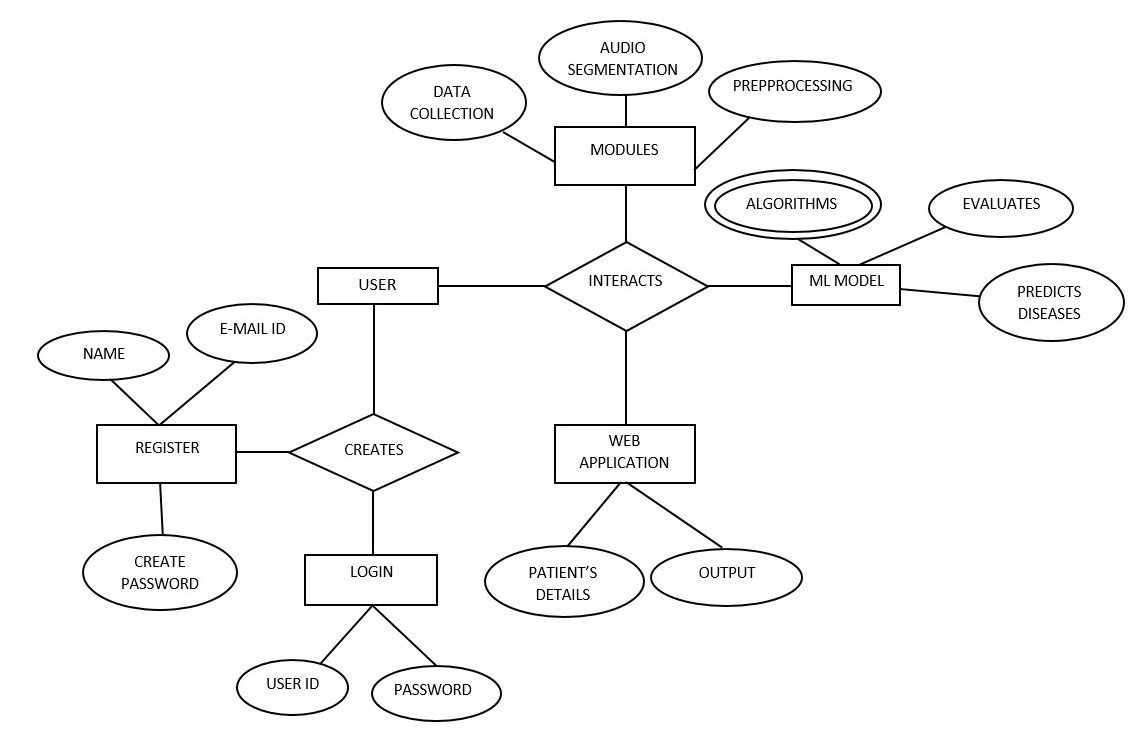
Werkzeug is Unicode aware and doesn’t enforce any dependencies. It is up to the developer to choose a template engine, database adapter, and even how to handle requests. It can be used to build all sorts of end user applications such as blogs, wikis, or bulletin boards.

Flask wraps Werkzeug, using it to handle the details of WSGI while providing more structure and patterns for defining powerful applications.

**CHAPTER 4**

**SYSTEM DESIGN**

**4.1 ER DIAGRAM**

****

**Fig 4.1 ER DIAGRAM**

**4.2 DATA DICTIONARY**

A data dictionary, or metadata repository, as defined in the IBM Dictionary of Computing, is a "centralized repository of information about data such as meaning, relationships to other data, origin, usage, and format". Oracle defines it as a collection of tables with metadata. The term can have one of several closely related meanings pertaining to databases and database management systems (DBMS):

* A document describing a database or collection of databases user
* An integral component of a DBMS that is required to determine its structure
* A piece of middleware that extends or supplants the native data dictionary of a DBMS.

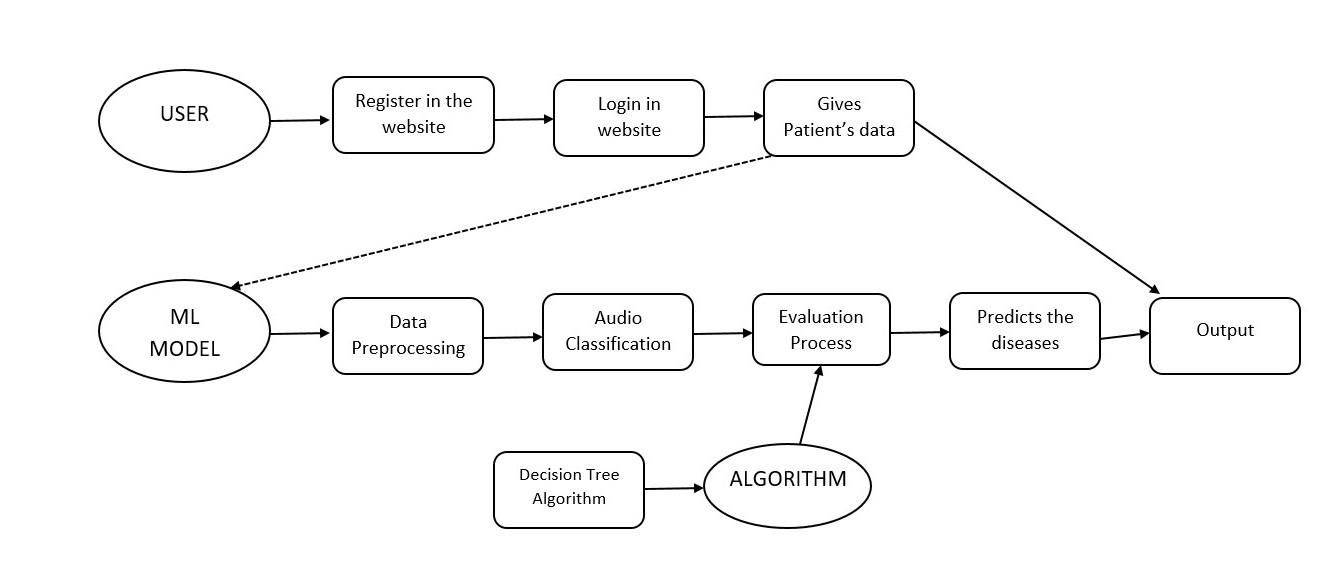
**4.2.1 User:**

|  |  |
| --- | --- |
| **FIELD** | **DATATYPE** |
| User\_id | ObjectId |
| Email\_id | String |
| Password | String |
| Age | Number |
| BMI | Number |
| Gender | Boolean |

**Table 4.1 Data Dictionary for user**

**4.3 DATA FLOW DIAGRAM**

A picture is worth a thousand words. A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system. It is usually beginning with a context diagram as the level 0 of the DFD diagram, a simple representation of the whole system. To elaborate further from that, we drill down to a level 1 diagram with lower level functions decomposed from the major functions of the system. This could continue to evolve to become a level 2 diagram when further analysis is required. Progression to level 3, 4 and so on is possible but anything beyond level 3 is not very common. Please bear in mind that the level of details for decomposing function really depends on the complexity that function.



**Fig 4.2 Data flow diagram**

**4.4 UML DIAGRAMS**

UML stands for Unified Modeling Language. It’s a rich language to model software solutions, application structures, system behavior and business processes.

There are 14 UML diagram types to help you model these behaviors. Unified

Modeling Language (UML) is a standard visual modeling language intended

to be used for

• modeling business and similar processes,

• analysis, design, and implementation of software-based systems

UML is a common language for business analysts, software architects and developers used to describe, specify, design, and document existing or new business processes, structure and behavior of artifacts of software systems.

Specification explained that process:

• provides guidance as to the order of a team’s activities,

• specifies what artifacts should be developed,

• directs the tasks of individual developers and the team as a whole, and

• offers criteria for monitoring and measuring a project’s products and

activities.

UML is intentionally process independent and could be applied in the context of different processes. Still, it is most suitable for use case driven, iterative and incremental development processes. An example of such process is Rational Unified Process (RUP).UML is not complete, and it is not completely visual. Given some UML diagram, we can't be sure to understand depicted part or behavior of the system from the diagram alone. Some information could be intentionally omitted from the diagram, some information represented on the diagram could have different interpretations, and some concepts of UML have no graphical notation at all, so there is no way to depict those on diagrams. For example, semantics of multiplicity of actors and multiplicity of use cases on use case diagrams is not defined precisely in the UML specification and could mean either concurrent or successive usage of use cases. Name of an abstract classifier is shown in italics while the final classifier has no specific graphical notation, so there is no way to determine whether the classifier is final or not from the diagram.

**4.4.1 List of UML Diagram Types**

So, what are the different UML diagram types? There are two main categories; structure diagrams and behavioral diagrams. Click on the links to learn more about a specific diagram type.

**4.4.2 Structure Diagrams**

Structure diagrams show the things in the modeled system. In a more technical term, they show different objects in a system. Behavioral diagrams show what should happen in a system. They describe how the objects interact with each other to create a functioning system.

**4.4.3 Class Diagram**

Class diagrams are the main building block of any object-oriented solution. It shows the classes in a system, attributes, and operations of each class and the relationship between each class. In most modeling tools, a class has three parts. Name at the top, attributes in the middle and operations or methods at the bottom. In a large system with many related classes, classes are grouped together to create class diagrams. Different relationships between classes are shown by different types of arrows.

**4.4.4 Component Diagram**

A component diagram displays the structural relationship of components of a software system. These are mostly used when working with complex systems with many components. Components communicate with each other using interfaces. The interfaces are linked using connectors. The image below shows a component diagram.

**4.4.5 Deployment Diagram**

A deployment diagram shows the hardware of your system and the software in that

hardware. Deployment diagrams are useful when your software solution is deployed across multiple machines with each having a unique configuration.

**4.4.6 Package Diagram**

As the name suggests, a package diagram shows the dependencies between different packages in a system. Check out this wiki article to learn more about the dependencies and elements found in package diagrams.

**4.4.7 Composite Structure Diagram**

As the name suggests, a package diagram shows the dependencies between different packages in a system. Check out this wiki article to learn more about the dependencies and elements found in package diagrams.

**4.4.8 Use Case Diagram**

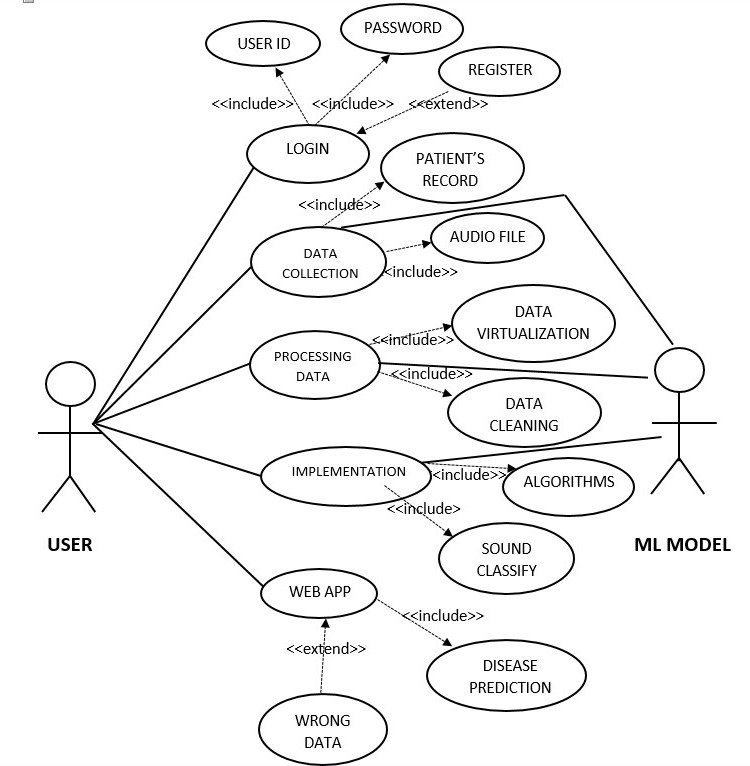
As the most known diagram type of the behavioral UML diagrams, use case diagrams give a graphic overview of the actors involved in a system, different functions needed by those actors and how these different functions interact. It’s a great starting point for any project discussion because you can easily identify the main actors involved and the main processes of the system. You can create use case diagrams using our tool and/or get started instantly using our use case templates.

**4.4.9 Activity Diagram**

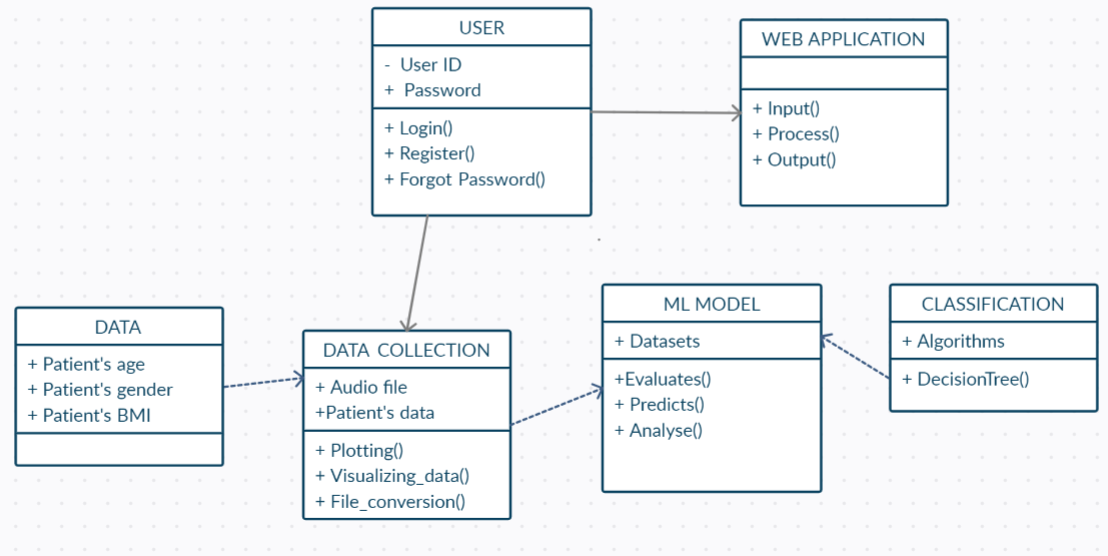
Activity diagrams represent workflows in a graphical way. They can be used to describe the business workflow or the operational workflow of any component in a system. Sometimes activity diagrams are used as an alternative to State machine diagrams. Check out this wiki article to learn about symbols and usage of activity diagrams.

**4.4.10 Sequence Diagram**

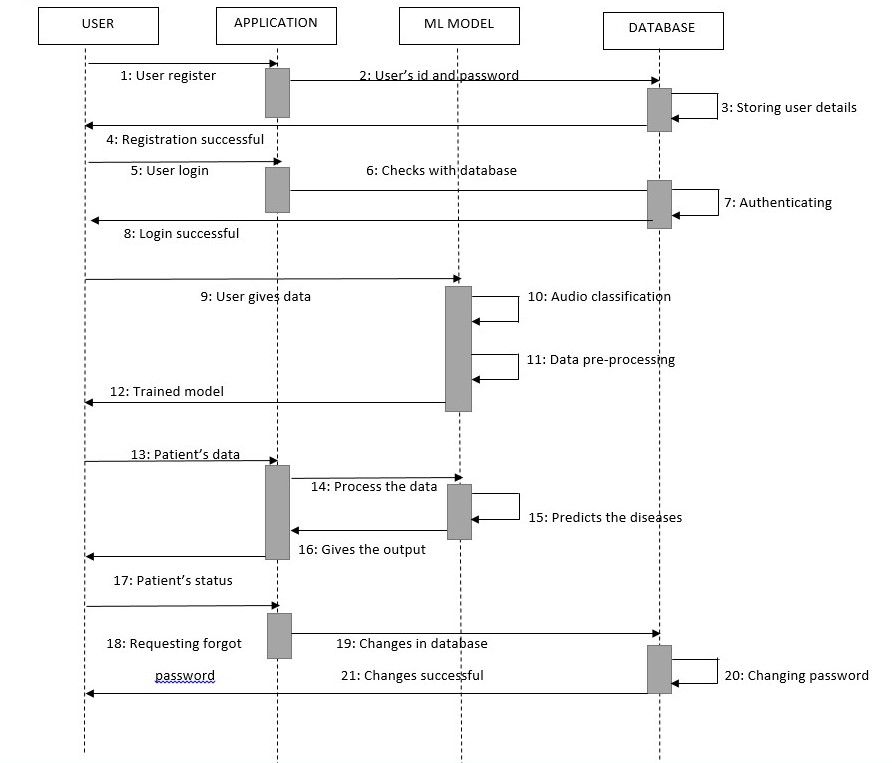
Sequence diagrams in UML show how objects interact with each other and the order those interactions occur. It’s important to note that they show the interactions for a scenario. The processes are represented vertically, and interactions are shown as arrows. This article explains the purpose and the basics of Sequence diagrams. Also, check out this complete Sequence Diagram Tutorial to learn more about sequence diagrams. You can also instantly start drawing using our sequence diagram templates.



**Fig 4.3: Use case diagram**



**Fig 4.4: Class diagram**



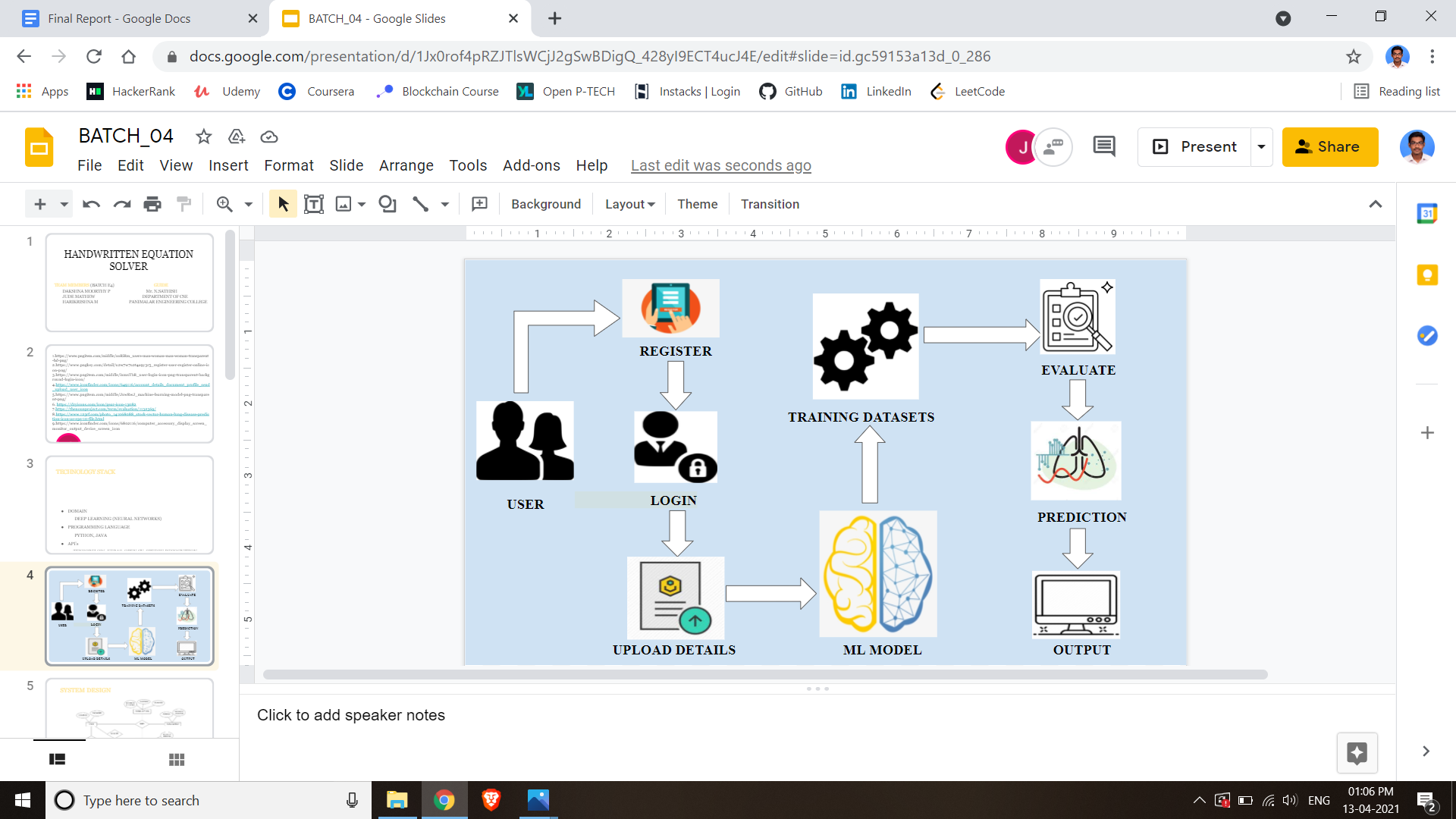
**Fig 4.5: Sequence diagram**

**CHAPTER 5**

**SYSTEM ARCHITECTURE**

**5.1 ARCHITECTURE OVERVIEW**

System architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behaviors of the system.



**Fig 5.1 System Architecture Diagram**

**5.2 SYSTEM MODULE**

The Sound Classification System contains different modules functions namely:

Module 1: Data collection and audio segmentation

Module 2: Preprocessing and cleaning

Module 3: Implementation of algorithm

Module 4: Respiratory disease classification

**5.2.1 MODULES EXPLANATION**

**5.2.1.1: Data collection and audio segmentation**

In this module the data is collected in the form of audio, these audio files are stored and they are converted in the form of a text file, this is how the data is collected. These data are collected from different patients' lungs and it is recorded. Data virtualization of audio and text files.

**5.2.1.2: Preprocessing and cleaning**

Data preprocessing involves the transformation of the raw dataset into an understandable format. Preprocessing data is a fundamental stage in data mining to improve data efficiency. The data preprocessing methods directly affect the outcomes of any analytic algorithm. These data are pre-processed and we have to clean the unwanted values all the null values.

**5.2.1.3: Implementation of algorithm**

An implementation is a realization of a technical specification or algorithm as a program, software component, or other computer system through computer programming and deployment. Once everything is set up, it is time to actually implement the algorithm. Implementation of various classification algorithms is done. With these algorithms we can easily classify the sounds easily.

**5.2.1.4: Respiratory disease classification**

In this module it will classify whether the person is healthy or infected. It gives either of these as a result. If the person is infected it will identify the type of disease and display the disease name else it will display the person is normal.

**5.3 PROGRAM LANGUAGE DESIGN**

**Disease Prediction Algorithm:**

**Step 1:** Reading the pre-processed data csv file.

**Step 2:** Define variables X and Y from dataset.

**Step 3:** Encoding categorical data e.g. gender as a dummy variable.

**Step 4:** Encoding categorical data e.g. disease outcome as a dummy variable.

**Step 5:** Splitting the dataset into the Training set and Test set.

**Step 6:** Fitting Classifier to the Training Set (Decision Tree Classifier)

**Step 7:** if request.method == 'POST':

A = request.form['Age']

N = request.form['Gender']

P = request.form['BMI']

**Step 8:** Creating a function for Annotation data for identifying recording\_info and recording\_annotations.

**Step 9:** Summed number of crackles / wheezes are normalized by the duration of the recording

duration = annotation.iloc[-1, 1] - annotation.iloc[0, 0]

info['Crackles'] = crackles/duration # crackles per second

info['Wheezes'] = wheezes/duration # wheezes per second

**Step 10:** prediction = classifier.predict(pred)

if prediction==5:

print('Bronchiolitis')

if prediction==4:

print('Pneumonia')

if prediction==3:

print('Bronchiectasis')

if prediction==2:

print('COPD')

if prediction==1:

print('Healthy')

if prediction==0:

print('URTI')

**5.3.1 Register**

The user launches the web application and register themselves by giving Email-id, username and creating password.

**5.3.2 User Login**

If user already a registered member they can directly login to the web application by entering Email-id and password.

**5.3.3 Data Preprocessing and Handling Missing Data**

* Reading the diagnosis data.
* Reading the demographic data.
* Calculating children BMI and adding that as a new column.
* Dropping the Height & Weight for Children columns as they are now replaced by BMI for Children.
* Adding disease labels to demographic data.
* Combine Adult and Children BMI together in a new df: demographicSummary.
* Review missing data in the demographic information.
* Eliminate rows with at least three missing values.
* Review rows with remaining missing values.
* Impute missing values for BMI based on the mean of similar rows.

**CHAPTER 6**

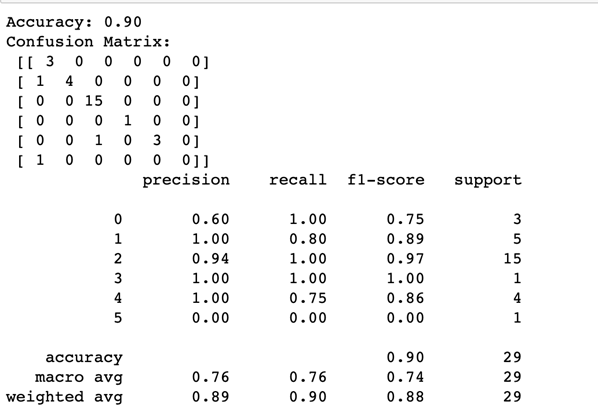
**TESTING**

**6.1 SYSTEM TESTING**

The testing and the evaluation part are the most important part in any research or development of a model. As we need to understand and check whether our model is performing the way it should be.

**Decision Tree**

The accuracy and the classification report is as follows.

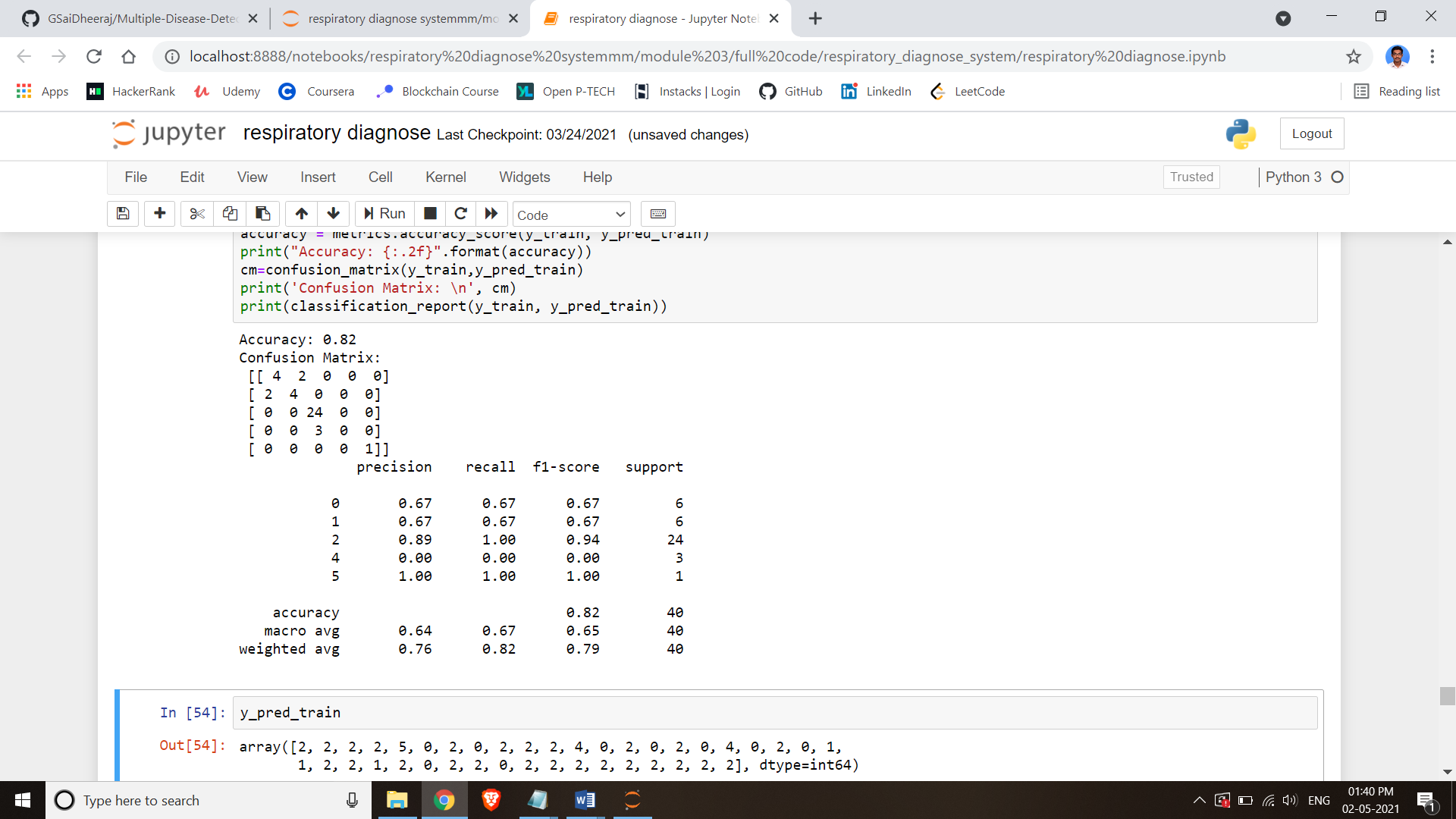


**Fig 6.1 Decision Tree Classifier Report**

In the above figure we can see the precision, recall and f1-score and the global accuracy of the model. The weighted average of the f1 has to be considered in classification problems rather than the accuracy. Here we get 88 percent as the weighted average and that shows a good performance for the algorithm.

**Support Vector Machine (SVM)**

The accuracy and the classification report is as follows.

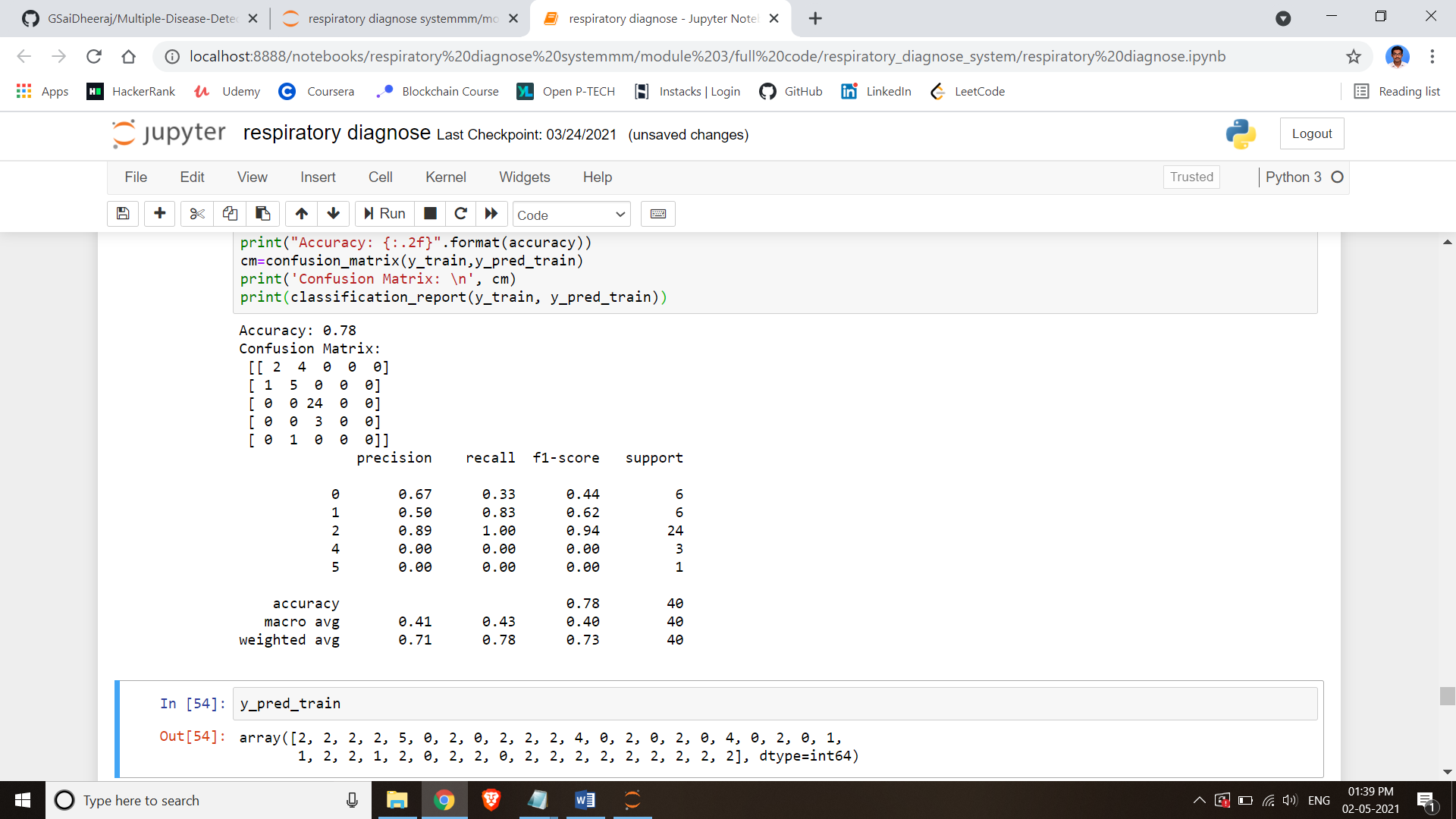


**Fig 6.2 Support Vector Machine Classifier Report**

In the support vector algorithm, we first import SVC which is a linear kernel that supports faster and better scale for prediction using sklearn.svm. The training set and testing set is divided 70 percent and 30 percent respectively and using fit () we call the data sets for training and testing. The SVM classifier has a random state set to 42 and since svc is being used kernel is linear. We use the metrics from the sklearn library to import the confusion matrix and the classification report for the algorithm.

**Logistic Regression**

In the Logistic Regression, we first import Logistic Regression which is a linear kernel that supports categorical data for prediction using sklearn.linear model. The training set and testing set is divided 70 percent and 30 percent respectively and using fit () we call the data sets for training and testing. The Logistic Regression classifier has a random state set to 42. After this point we predict the accuracy and find the confusion matric and the classification report.



**Fig 6.3 Logistic Regression Classifier Report**

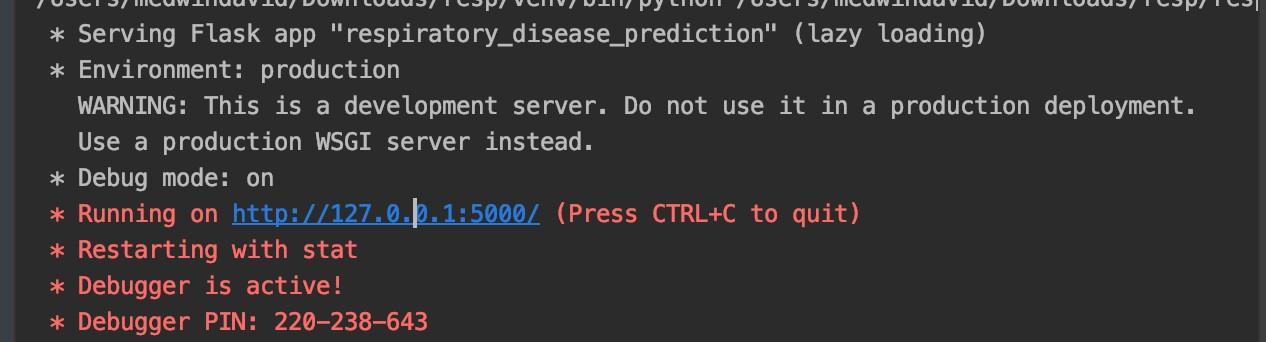
**CNN with VGG16**

In the CNN model we get validation accuracy of 82 percent and validation loss of 1.13, here after a certain number of epochs we can see that the validation loss starts to increase back. This results in over fitting. We will not be proceeding with CNN model because the model stops learning at a particular point and because of over fitting we could say the model is not a perfect choice.

#### 

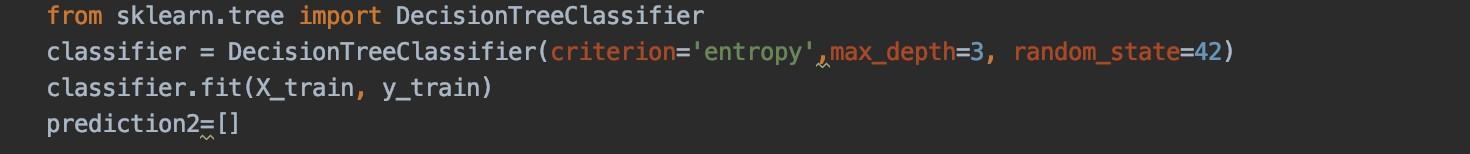
#### **6.2 Web Application Testing**

We use the flask framework for developing the webapp, along with CSS and html. On starting the application, it is bonded to a default address <http://127.0.0.1:5000/>



**Fig 6.4: Web Application Testing**

On entering the diseases prediction page, the best performing algorithm, Decision tree is called to do the classification for the input data. The input is then validated with an algorithm and the training data from our model and predicts the result for the input given by the user.



**Fig 6.5: web application testing inputs**

**6.3 Unit Testing**

In computer programming, unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are short code fragments created by programmers or occasionally by white box testers during the development process. Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended.

**6.4 TEST CASES**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test  Case Id | Test  Cases | Priority | Input  Test  Data | Test Case Description | Expected Results | Actual  Results | Pass/  Fail |
| TU0  1 | User  registration | A | Enter  Email-id  and  Password. | Check user with the  database | User  should be  available in database | User  present in database | Pass |
| TU0  2 | Machine Learning model | A | Patient’s diagnosis data | Splitting up the data for training and testing | Training with good accuracy | Trained Machine learning model | Pass |
| TU0  3 | Upload  med.details | A | Enter  med.details | Update  med.details | Should upload  success  fully | Details are stored  successfully | Pass |
| TU0  4 | Upload  audio files | A | Path wave  Extension file | Considered as ML model training datasets | Successfully uploaded the audio file | Correctly uploaded the audio file | Pass |
| TU0  5 | Type of  disease | A | Disease  Name display | Disease  specification | Name of  disease | Patient  condition | Pass |

**CHAPTER 7**

**CONCLUSION AND FUTURE ENHANCEMENT**

**7.1 CONCLUSION**

In our project, we have predicted respiratory diseases from the respiratory sounds database using decision trees. Our work comprised of the comparison with machine learning algorithm. Our project mainly consists of three areas, preprocessing of the data, prediction of diseases, and developing the interface for users to use. In the preprocessing, we are handling the missing data, normalizing the values and eliminating any unwanted data from our dataset and creating a new preprocessed data. The prediction with decision trees gives an accuracy rate of 90 percent. CNN models can be used when there is a large amount of data is available to train, when learning with less data the CNN model is not as appropriate as over fitting occurs in the training of model. The more the accuracy and f1 average weight of the algorithms the model’s predictions become accurate.

**7.2 FUTURE ENHANCEMENT**

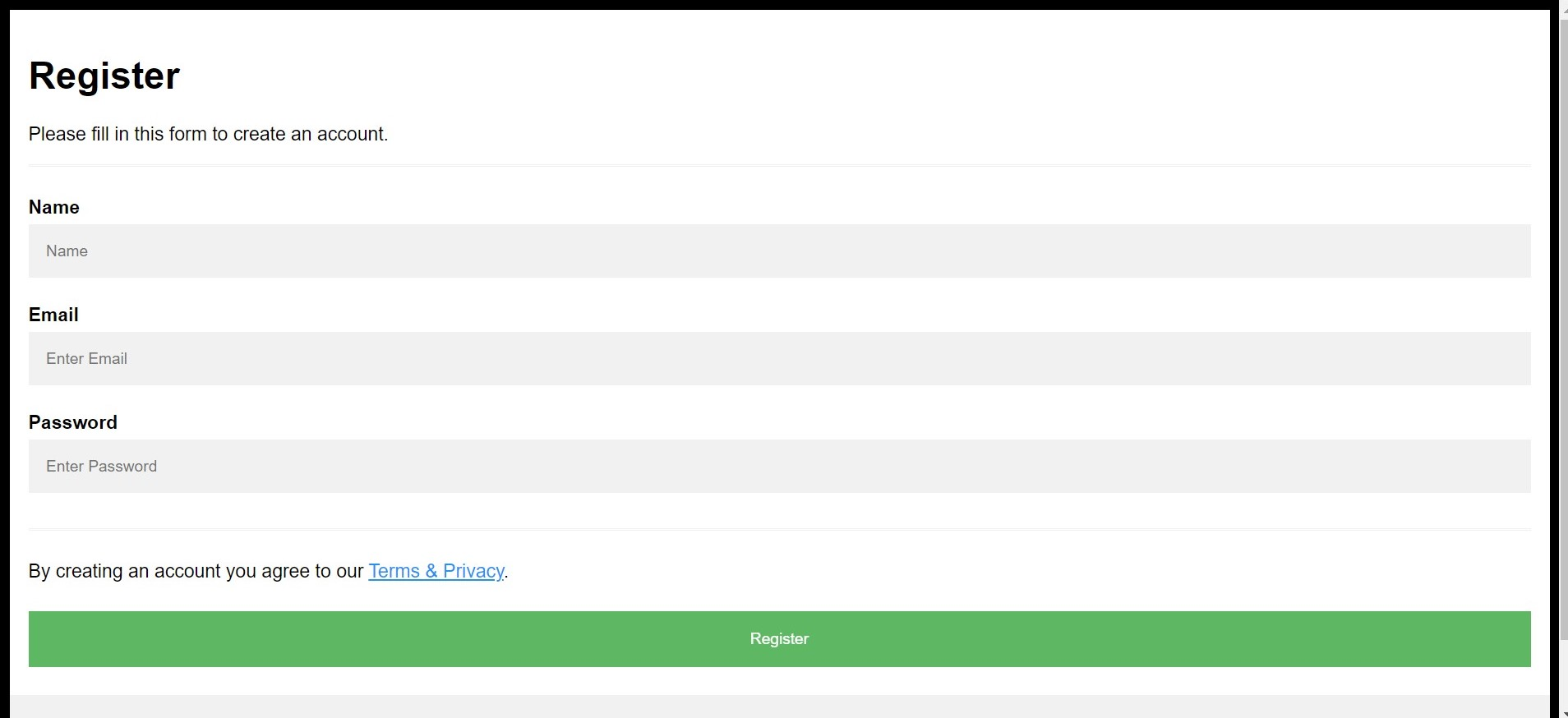
The future work of the application consists of mostly collecting more data and trying to implement with the CNN model, also instead of manual annotation of the audio files teach the model to automatically annotate the recordings. This web application can add storage functionalities where the users can access their previous breath sound checks and also the automatic annotation process of sounds which helps the users to easily identify the disease. A desktop or mobile application can also be built to make the process easier for the users.

**APPENDICES**

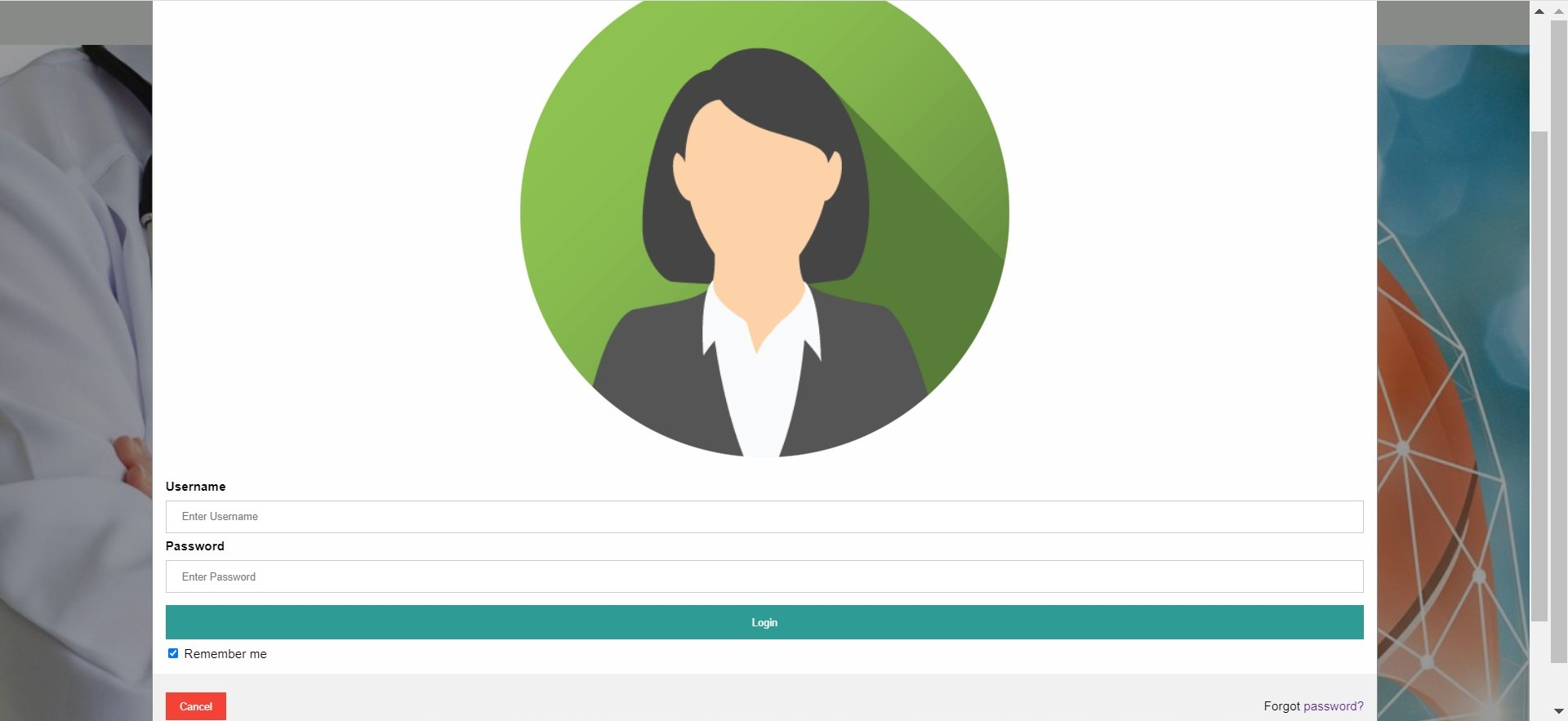
**A 1. SAMPLE SCREENS**



**Fig 1: Home Page**



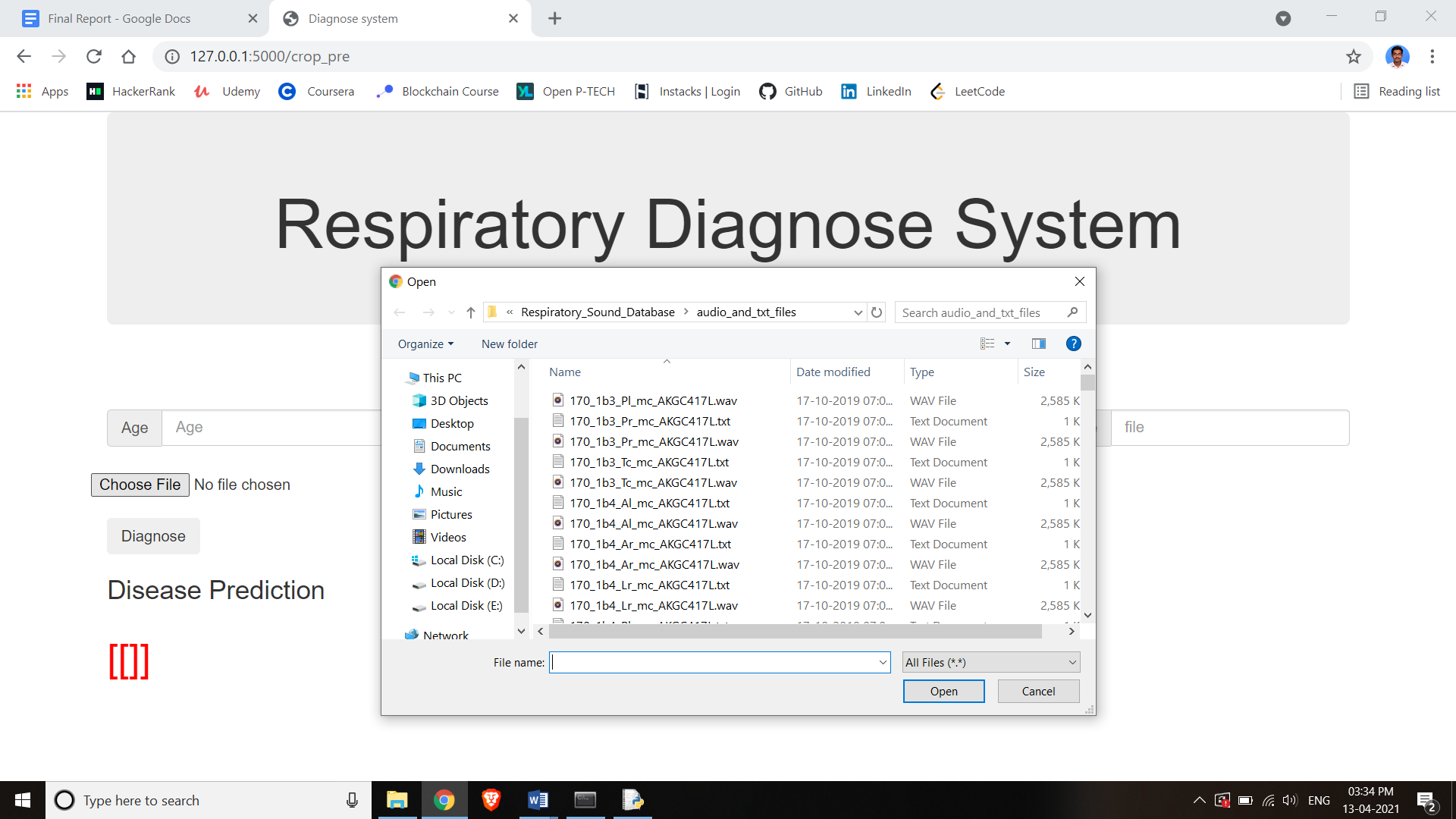
**Fig 2: Register Page**



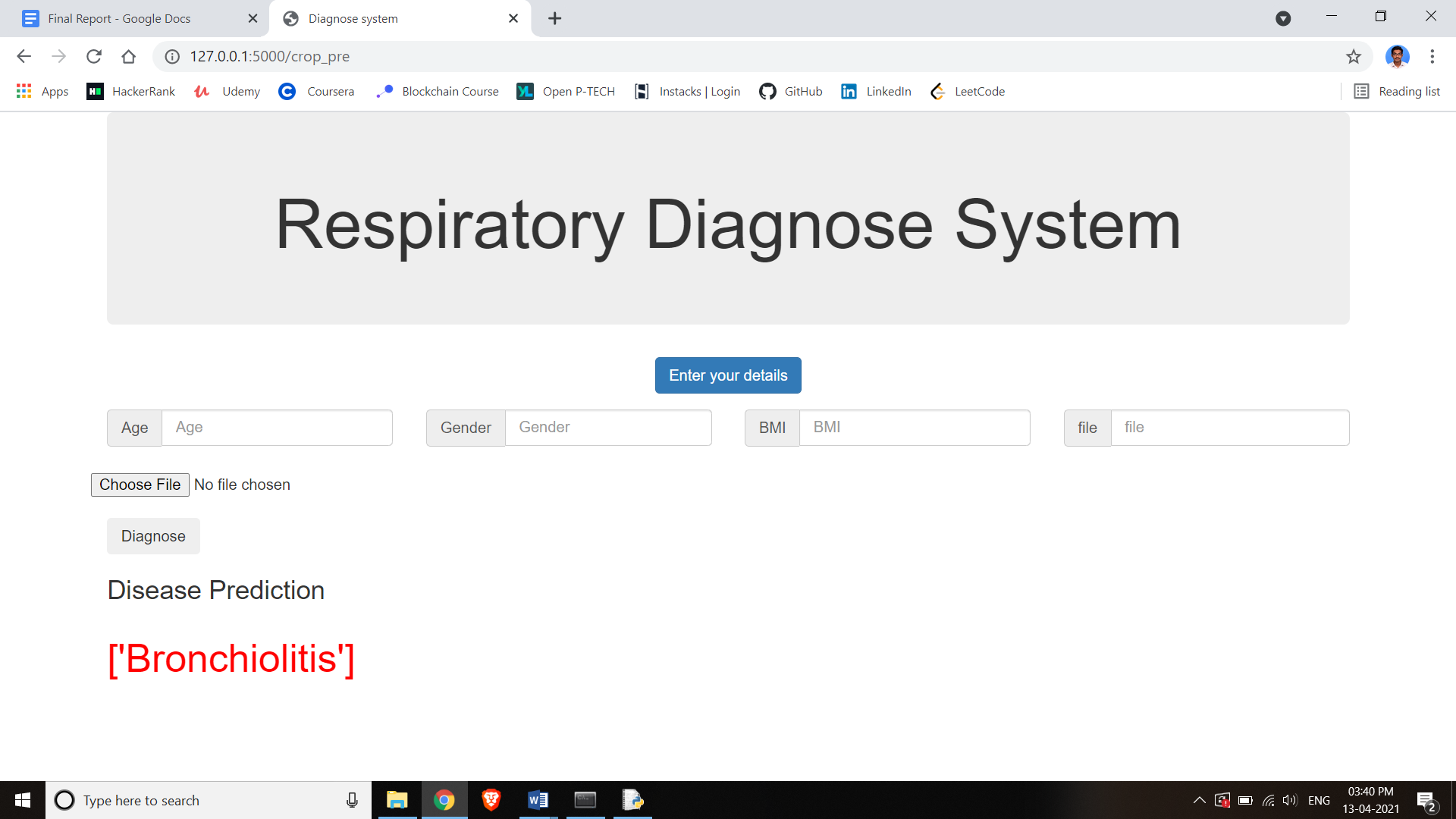
**Fig 3: Login Page**



**Fig 4: Disease Prediction using Wheezes and Crackles**



**Fig 5: Uploading Audio file for Prediction**



**Fig 6: Disease Prediction using Audio file**

**A2. SAMPLE CODE**

**PYTHON**

**Respiratory Disease Prediction.py**

from flask import Flask ,render\_template,request,jsonify,session

from flask import Flask,abort,render\_template,request,redirect,url\_for

from werkzeug import secure\_filename

import os

import sqlite3 as sql

import base64

import pandas as pd

from sklearn.preprocessing import LabelEncoder

#from flask\_bootstrap import Bootstrap

import numpy as np

from sklearn.utils import shuffle

app = Flask(\_\_name\_\_)

app.secret\_key = 'any random string'

UPLOAD\_FOLDER = '/tmp'

app.config['UPLOAD\_FOLDER'] = UPLOAD\_FOLDER

@app.route('/', methods=['GET', 'POST'])

def home():

return render\_template('index.html')

def validate(username,password):

con = sql.connect('static/chat.db')

completion = False

with con:

cur = con.cursor()

cur.execute('SELECT \* FROM persons')

rows = cur.fetchall()

for row in rows:

dbuser = row[1]

dbpass = row[2]

if dbuser == username:

completion = (dbpass == password)

return completion

@app.route('/login', methods=['GET', 'POST'])

def login():

error = None

if request.method == 'POST':

username = request.form['username']

password = request.form['password']

completion = validate(username,password)

if completion == False:

error = 'invalid Credentials. please try again.'

else:

session['username'] = request.form['username']

return render\_template('index111.html')

return render\_template('index111.html', error=error)

@app.route('/view', methods=['GET', 'POST'])

def view():

return render\_template('index111.html')

@app.route('/register', methods = ['GET','POST'])

def register():

if request.method == 'POST':

try:

name = request.form['name']

username = request.form['username']

password = request.form['password']

with sql.connect("static/chat.db") as con:

cur = con.cursor()

cur.execute("INSERT INTO persons(name,username,password) VALUES (?,?,?)",(name,username,password))

con.commit()

msg = "Record successfully added"

except:

con.rollback()

msg = "error in insert operation"

finally:

return render\_template("index.html",msg = msg)

con.close()

return render\_template('register.html')

@app.route('/list')

def list():

con = sql.connect("static/chat.db")

con.row\_factory = sql.Row

cur = con.cursor()

cur.execute("select \* from persons")

rows = cur.fetchall();

return render\_template("list.html",rows = rows)

@app.route('/crop\_predict',methods=['GET','POST'])

def crop():

dataset = pd.read\_csv('dataset/preprocess\_dataset.csv')

X = dataset.iloc[:,[1,2,3,5,6]].values

y = dataset.iloc[:,4].values

# encoding categorical data e.g. gender as a dummy variable

from sklearn.preprocessing import LabelEncoder

labelencoder\_X = LabelEncoder()

X[:,1] = labelencoder\_X.fit\_transform(X[:,1])

# encoding categorical data e.g. disease outcome as a dummy variable

y,class\_names = pd.factorize(y)

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

# Fitting Classifier to the Training Set

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion='entropy',max\_depth=3, random\_state=42)

classifier.fit(X\_train, y\_train)

prediction2=[]

if request.method == 'POST':

A = request.form['Age']

N = request.form['Gender']

P = request.form['BMI']

K = request.form['crackles']

ph = request.form['wheezes']

print(A,N,P,K,ph)

columns = ['A','N','P','K','ph']

values = np.array([A,N,P,K,ph])

pred = pd.DataFrame(values.reshape(-1, len(values)),columns=columns)

prediction = classifier.predict(pred)

if prediction==5:

print('Bronchiolitis')

prediction1='Bronchiolitis'

if prediction==4:

print('Pneumonia')

prediction1='Pneumonia'

if prediction==3:

print('Bronchiectasis')

prediction1='Bronchiectasis'

if prediction==2:

print('COPD')

prediction1='COPD'

if prediction==1:

print('Healthy')

prediction1='Healthy'

if prediction==0:

print('URTI')

prediction1='URTI'

prediction2.append(prediction1)

return render\_template('crop.html',predict=prediction2,display=True)

**Respiratory Diagnose.ipynb**

import pandas as pd

import numpy as np

import os

import matplotlib.pyplot as plt

# Don't Show Warning Messages

import warnings

warnings.filterwarnings('ignore')

#view the respiratory dataset

os.listdir('C:/files/respiratory\_sound\_database')

# view the Respiratory\_Sound\_Database

os.listdir('C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database')

# Play an audio file

from pydub import AudioSegment

import IPython

# We will listen to this file:

# 213\_1p5\_Pr\_mc\_AKGC417L.wav

audio\_file = '175\_1b1\_Pl\_sc\_Litt3200.wav'

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files/' + audio\_file

IPython.display.Audio(path)

#demographic\_info.txt

path ='C:/files/respiratory\_sound\_database/demographic\_info.txt'

col\_names = ['patient\_id', 'age', 'sex', 'adult\_bmi', 'child\_weight', 'child\_height']

# Adult BMI (kg/m2)

# Child Weight (kg)

# Child Height (cm)

df\_demo = pd.read\_csv(path, sep=" ", header=None, names=col\_names)

df\_demo.head(10)

#patient\_diagnosis.csv

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/patient\_diagnosis.csv'

df\_diag = pd.read\_csv(path, header=None, names=['patient\_id', 'diagnosis'])

df\_diag.head(10)

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/filename\_differences.txt'

df\_diff = pd.read\_csv(path, sep=" ", header=None, names=['file\_names'])

df\_diff.head(10)

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/filename\_format.txt'

data = open(path, 'r').read()

data

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files'

os.listdir(path)

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files/176\_1b3\_Pl\_mc\_AKGC417L.txt'

col\_names = ['Beginning\_of\_respiratory\_cycle', 'End\_of\_respiratory\_cycle', 'Presence/absence\_of\_crackles', 'Presence/absence\_of\_wheezes']

# Respiratory cycle column values are in 'seconds'.

# Presence = 1

# Absence = 0

df\_annot = pd.read\_csv(path, sep="\t", header=None, names=col\_names)

df\_annot.head(10)

import soundfile as sf

# Define helper functions

# Load a .wav file.

# These are 24 bit files. The PySoundFile library is able to read 24 bit files.

# https://pysoundfile.readthedocs.io/en/0.9.0/

def get\_wav\_info(wav\_file):

data, rate = sf.read(wav\_file)

return data, rate

# source: Andrew Ng Deep Learning Specialization, Course 5

def graph\_spectrogram(wav\_file):

data, rate = get\_wav\_info(wav\_file)

nfft = 200 # Length of each window segment

fs = 8000 # Sampling frequencies

noverlap = 120 # Overlap between windows

nchannels = data.ndim

if nchannels == 1:

pxx, freqs, bins, im = plt.specgram(data, nfft, fs, noverlap = noverlap)

elif nchannels == 2:

pxx, freqs, bins, im = plt.specgram(data[:,0], nfft, fs, noverlap = noverlap)

return pxx

"""# create spectogram"""

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files/176\_1b3\_Pl\_mc\_AKGC417L.wav'

x = graph\_spectrogram(path)

"""# Read an audio file as a numpy array"""

# choose an audio file

audio\_file = '176\_1b3\_Pl\_mc\_AKGC417L.wav'

path = \

'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files/' + audio\_file

# read the file

data, rate = sf.read(path)

print(data,rate)

# display the numpy array

data1=pd.DataFrame(data)

#data1.to\_csv("a.csv")

"""# slice a section from an audio file"""

from pydub import AudioSegment

# note: Time is given in seconds. Will be converted to milliseconds later.

start\_time = 0

end\_time = 7

t1 = start\_time \* 1000 # pydub works in milliseconds

t2 = end\_time \* 1000

newAudio = AudioSegment.from\_wav(path) # path is defined above

newAudio = newAudio[t1:t2]

newAudio.export('new\_slice.wav', format="wav")

# Lets listen to the new slice

IPython.display.Audio('new\_slice.wav')

path = \

'new\_slice.wav'

x = graph\_spectrogram(path)

# choose an audio file

audio\_file = 'new\_slice.wav'

import glob

path = r'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files'

all\_files = glob.glob(path + "/\*.txt")

def AnnotationData(filename, path):

"""

For each filename in a given path, extracts the information from the file naming convention and reads the information

within the file.

Returns two dataframes: one with information on file naming, other with information from the file itself.

"""

words = filename[len(path):-4].split('\_')

recording\_info = pd.DataFrame(data = [words], columns = ['Patient number', 'Recording index', 'Chest location','Acquisition mode','Recording equipment'])

recording\_annotations = pd.read\_csv(filename, names = ['t\_start', 't\_end', 'Crackles', 'Wheezes'], delim\_whitespace=True)

return (recording\_info, recording\_annotations)

# Calling above function to read annotation files

infoList = []

for filename in all\_files:

(info, annotation) = AnnotationData(filename, path)

# Number of crackles / wheezes for all respiratory cycles within each recording are summed

crackles = annotation['Crackles'].sum()

wheezes = annotation['Wheezes'].sum()

# Summed number of crackles / wheezes are normalized by the duration of the recording

duration = annotation.iloc[-1, 1] - annotation.iloc[0, 0]

info['Crackles'] = crackles/duration # crackles per second

info['Wheezes'] = wheezes/duration # wheezes per second

infoList.append(info)

infoList

# Converting infoList to dataframe

info = pd.concat(infoList, axis = 0).sort\_values(by="Patient number")

# As there are several recordings for some patients, the averages of normalized number of crackles and wheezes are taken for each patient

crackles\_wheezes = info.iloc[:, [0,5,6]].groupby("Patient number").mean().reset\_index()

# Combining crackles\_wheezes with demographics summary data to input into Decision Tree

dem\_crackles\_wheezes = pd.concat([demographicSummary, crackles\_wheezes.iloc[:,[1,2]]], axis=1, sort=False).dropna()

"""# Decision Tree Algorithm"""

dataset = dem\_crackles\_wheezes

print(dataset.head())

# Dropping asthma and LRTI cases as these have only 1 patient

dataset = dataset.drop(dataset[ (dataset.Disease == 'Asthma') | (dataset.Disease == 'LRTI') ].index).reset\_index(drop=True)

dataset.head(10)

dataset.to\_csv("preprocess\_dataset.csv")

X = dataset.iloc[:,[1,2,3,5,6]].values

y = dataset.iloc[:,4].values

# encoding categorical data e.g. gender as a dummy variable

from sklearn.preprocessing import LabelEncoder

labelencoder\_X = LabelEncoder()

X[:,1] = labelencoder\_X.fit\_transform(X[:,1])

# encoding categorical data e.g. disease outcome as a dummy variable

y,class\_names = pd.factorize(y)

print(class\_names)

#y

# Splitting the dataset into the Training set and Test set

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size = 0.25, random\_state = 42)

#len(X\_train)

X\_test

#len(y\_train)

# Fitting Classifier to the Training Set

from sklearn.tree import DecisionTreeClassifier

classifier = DecisionTreeClassifier(criterion='entropy',max\_depth=3, random\_state=42)

classifier.fit(X\_train, y\_train)

# Model performance on training set

y\_pred\_train =classifier.predict(X\_train)

from sklearn import metrics

from sklearn.metrics import confusion\_matrix, classification\_report

accuracy = metrics.accuracy\_score(y\_train, y\_pred\_train)

print("Accuracy: {:.2f}".format(accuracy))

cm=confusion\_matrix(y\_train,y\_pred\_train)

print('Confusion Matrix: \n', cm)

print(classification\_report(y\_train, y\_pred\_train))

#y\_pred\_train

values = np.array([[3.0, 1, 30.741351682533697, 10.06535720983032765,

0.036425876596688465]])

y\_pred=classifier.predict(values)

y\_pred

"""# TESTING PROCESS"""

AGE=float(input("enter your age"))

GENDER=float(input("enter your gender"))

BMI=float(input("enter your BMI"))

file1=str(input("enter yourfile name"))

import glob

path = r'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files'

file = r'C:/files/respiratory\_sound\_database/Respiratory\_Sound\_Database/audio\_and\_txt\_files/'+file1+'.txt'

all\_files = glob.glob(file)

def AnnotationData(filename, path):

words = filename[len(path):-4].split('\_')

recording\_info = pd.DataFrame(data = [words], columns = ['Patient number', 'Recording index', 'Chest location','Acquisition mode','Recording equipment'])

recording\_annotations = pd.read\_csv(filename, names = ['t\_start', 't\_end', 'Crackles', 'Wheezes'], delim\_whitespace=True)

return (recording\_info, recording\_annotations)

# Calling above function to read annotation files

infoList = []

for filename in all\_files:

(info, annotation) = AnnotationData(filename, path)

# Number of crackles / wheezes for all respiratory cycles within each recording are summed

crackles = annotation['Crackles'].sum()

wheezes = annotation['Wheezes'].sum()

# Summed number of crackles / wheezes are normalized by the duration of the recording

duration = annotation.iloc[-1, 1] - annotation.iloc[0, 0]

info['Crackles'] = crackles/duration # crackles per second

info['Wheezes'] = wheezes/duration # wheezes per second

infoList.append(info)

CRACKLES=float(info['Crackles'])

WHEEZES=float(info['Wheezes'])

values = np.array([[AGE,GENDER,BMI,CRACKLES,WHEEZES]])

print(values)

prediction=classifier.predict(values)

prediction

if prediction==5:

print('Bronchiolitis')

if prediction==4:

print('Pneumonia')

if prediction==3:

print('Bronchiectasis')

if prediction==2:

print('COPD')

if prediction==1:

print('Healthy')

if prediction==0:

print('URTI')

**HTML AND CSS CODE:**

**1. HOME PAGE: index.html**

<head>

<title>Home page</title>

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" type="text/css" href="aa/aa.css">

<link rel="stylesheet" type="text/css" href="aa/an.js">

<style>

body {font-family: Arial, Helvetica, sans-serif;}

/\* Full-width input fields \*/

input[type=text], input[type=password] {

width: 100%;

padding: 12px 20px;

margin: 8px 0;

display: inline-block;

border: 1px solid #ccc;

box-sizing: border-box;

}

/\* Set a style for all buttons \*/

button {

background-color: #2e9c95;

color: white;

padding: 14px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

}

button:hover {

opacity: 0.8;

}

/\* Extra styles for the cancel button \*/

.cancelbtn {

width: auto;

padding: 10px 18px;

background-color: #f44336;

}

/\* Center the image and position the close button \*/

.imgcontainer {

text-align: center;

margin: 24px 0 12px 0;

position: relative;

}

img.avatar {

width: 40%;

border-radius: 50%;

}

.container {

padding: 16px;

}

span.psw {

float: right;

padding-top: 16px;

}

/\* The Modal (background) \*/

.modal {

display: none; /\* Hidden by default \*/

position: fixed; /\* Stay in place \*/

z-index: 1; /\* Sit on top \*/

left: 0;

top: 0;

width: 100%; /\* Full width \*/

height: 100%; /\* Full height \*/

overflow: auto; /\* Enable scroll if needed \*/

background-color: rgb(0,0,0); /\* Fallback color \*/

background-color: rgba(0,0,0,0.4); /\* Black w/ opacity \*/

padding-top: 60px;

}

/\* Modal Content/Box \*/

.modal-content {

background-color: #fefefe;

margin: 5% auto 15% auto; /\* 5% from the top, 15% from the bottom and centered \*/

border: 1px solid #888;

width: 80%; /\* Could be more or less, depending on screen size \*/

}

/\* The Close Button (x) \*/

.close {

position: absolute;

right: 25px;

top: 0;

color: #000;

font-size: 35px;

font-weight: bold;

}

.close:hover,

.close:focus {

color: red;

cursor: pointer;

}

/\* Add Zoom Animation \*/

.animate {

-webkit-animation: animatezoom 0.6s;

animation: animatezoom 0.6s

}

@-webkit-keyframes animatezoom {

from {-webkit-transform: scale(0)}

to {-webkit-transform: scale(1)}

}

@keyframes animatezoom {

from {transform: scale(0)}

to {transform: scale(1)}

}

/\* Change styles for span and cancel button on extra small screens \*/

@media screen and (max-width: 300px) {

span.psw {

display: block;

float: none;

}

.cancelbtn {

width: 100%;

}

}

body, html {

height: 100%;

margin: 0;

}

.bg {

/\* The image used \*/

background-image: url("{{url\_for('static', filename = 'resp.jpg')}}");

/\* Full height \*/

height: 100%;

/\* Center and scale the image nicely \*/

background-repeat: no-repeat;

background-size: cover;

}

</style>

<STYLE>A {text-decoration: none;} </STYLE>

</head>

<body link="white">

<marquee behavior="scroll" direction="left" onmouseover="this.stop();" onmouseout="this.start();" style="background-color:#e1e6e0;font-size: xxx-large;" >Respiratory Diagnose System</marquee>

<div class="bg" >

<center> </br></br></br></br></br></br>

<button onclick="document.getElementById('id01').style.display='block'" style="width:auto;">Login</button>

<button onclick="document.getElementById('').style.display='block'" style="width:auto;"><a href = "{{ url\_for('register')}}" >Register</a></button>

</center>

</div>

<div id="id01" class="modal">

<form class="modal-content animate" action="/login" method="post">

<div class="imgcontainer">

<span onclick="document.getElementById('id01').style.display='none'" class="close" title="Close Modal">&times;</span>

<img src="{{url\_for('static', filename = 'img\_avatar2.png')}}" alt="Avatar" class="avatar">

</div>

<div class="container">

<label for="uname"><b>Username</b></label>

<input type="text" placeholder="Enter Username" name="username" required value="{{request.form.username }}">

<label for="psw"><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="password" required value="{{request.form.password }}">

<button type="submit">Login</button>

<label>

<input type="checkbox" checked="checked" name="remember"> Remember me

</label>

</div>

<div class="container" style="background-color:#f1f1f1">

<button type="button" onclick="document.getElementById('id01').style.display='none'" class="cancelbtn">Cancel</button>

<span class="psw">Forgot <a href="#">password?</a></span>

</div>

</form>

{% if error %}

<p class="error"><strong>Error:</strong> {{ error }}

{% endif %} </div>

<script>

// Get the modal

var modal = document.getElementById('id01');

var modal = document.getElementById('id02');

// When the user clicks anywhere outside of the modal, close it

window.onclick = function(event) {

if (event.target == modal) {

modal.style.display = "none";

}

}

</script>

</body>

</html>

**index111.html**

<html lang="en">

<head>

<meta charset="UTF-8">

<title> CRS </title>

<link rel="stylesheet" type="text/css" href="static/css/styles.css"/>

</head>

<body>

<div class="div1">

<p> Diagnose System </p>

</div>

<div class="div3">

<img src="resp1.jpg"/>

<div class="center"> Diagnose respiratory disease</div>

<ul><li>

<a href="{{ url\_for('crop')}}"> disease prediction </a> </li>

<li> <a href="{{ url\_for('crop1')}}"> Diagnose respiratory disease </a> </li></ul>

</div>

<div id="div4">

<p>@Copyrights</p>

</div>

</body>

</html>

**2. REGISTER PAGE: register.html**

<!DOCTYPE html>

<html>

<head>

<meta name="viewport" content="width=device-width, initial-scale=1">

<style>

body {

font-family: Arial, Helvetica, sans-serif;

background-color: black;

}

\* {

box-sizing: border-box;

}

/\* Add padding to containers \*/

.container {

padding: 16px;

background-color: white;

}

/\* Full-width input fields \*/

input[type=text], input[type=password] {

width: 100%;

padding: 15px;

margin: 5px 0 22px 0;

display: inline-block;

border: none;

background: #f1f1f1;

}

input[type=text]:focus, input[type=password]:focus {

background-color: #ddd;

outline: none;

}

/\* Overwrite default styles of hr \*/

hr {

border: 1px solid #f1f1f1;

margin-bottom: 25px;

}

/\* Set a style for the submit button \*/

.registerbtn {

background-color: #4CAF50;

color: white;

padding: 16px 20px;

margin: 8px 0;

border: none;

cursor: pointer;

width: 100%;

opacity: 0.9;

}

.registerbtn:hover {

opacity: 1;

}

/\* Add a blue text color to links \*/

a {

color: dodgerblue;

}

/\* Set a grey background color and center the text of the "sign in" section \*/

.signin {

background-color: #f1f1f1;

text-align: center;

}

</style>

</head>

<body>

<form action="{{url\_for('register')}}" method = "POST">

<div class="container">

<h1>Register</h1>

<p>Please fill in this form to create an account.</p>

<hr>

<label for="name"><b>Name</b></label>

<input type="text" placeholder="Name" name="name" required>

<label for="email"><b>Email</b></label>

<input type="text" placeholder="Enter Email" name="username" required>

<label for="password"><b>Password</b></label>

<input type="password" placeholder="Enter Password" name="password" required>

<hr>

<p>By creating an account you agree to our <a href="#">Terms & Privacy</a>.</p>

<button type="submit" class="registerbtn">Register</button>

</div>

<div class="container signin">

<p>Already have an account? <a href="{{ url\_for('login')}}">Sign in</a>.</p>

</div>

</form>

</body>

</html>

**3. DISEASE PREDICTION PAGE: crop.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Diagnose system</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<link rel="stylesheet" href="../static/css/autocomplete.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body bgcolor="#E6E6FA">

<div class="container">

<div class="jumbotron" style="text-align:center">

<h1>Respiratory Diagnose System </h1>

</div>

<div class="row">

<div class ="col-sm-12 col-centered" style="padding-bottom: 1em">

<center>

<div class="btn-group center">

<button id="auto\_btn" type="button" class="btn btn-primary" onclick="getData()">Enter your details </button>

</div>

</center>

</div>

</div>

<form action="/crop\_predict" method="POST" autocomplete="off">

<div class="row" style="padding-bottom: 0.5em , margin-left -30px" id="input-value">

<div class = "col-md-3">

<div class="input-group">

<span class="input-group-addon">Age</i></span>

<input type="text" class="form-control" name="Age" placeholder="Age">

</div>

</div>

<div class = "col-md-3">

<div class="input-group">

<span class="input-group-addon">Gender</i></span>

<input type="text" class="form-control" name="Gender" placeholder="Gender">

</div>

</div>

<div class ="col-md-3">

<div class="input-group">

<span class="input-group-addon">BMI</i></span>

<input type="text" class="form-control" name="BMI" placeholder="BMI">

</div>

</div>

<div class ="col-md-3">

<div class="input-group">

<span class="input-group-addon">crackles</i></span>

<input type="text" class="form-control" name="crackles" placeholder="crackles">

</div>

</div>

<div class = "col-md-3">

<div class="input-group">

<span class="input-group-addon">wheezes</i></span>

<input type="text" class="form-control" name="wheezes" placeholder="wheezes">

</div>

</div>

</div>

<div class="col-md-12" class="">

<br>

<input type="submit" class="btn" value="Diagnose">

</div>

</form>

<div class="col-md-6">

<h3>Disease Prediction </h3>

{% if display == True %}

<div class="table-responsive">

<table class="table table-hover">

<tbody>

<tr>

<h2 style="color:red">{{predict}}</h3>

</tr>

</tbody>

</table>

</div>

{% endif%}

</div>

</div>

<script src="../static/js/autocomplete1.js"></script>

<script type="text/javascript">

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

city.style.display = "block"

npk.style.display = "none"

function getData(){

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

// console.log(te.value);

npk.style.display = "block";

city.style.display = "none";

}

function citywise(){

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

// console.log(te.value);

npk.style.display = "none";

city.style.display = "block";

}

</script>

</body>

</html>

**crop1.html**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Diagnose system</title>

<meta charset="utf-8">

<meta name="viewport" content="width=device-width, initial-scale=1">

<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css">

<link rel="stylesheet" href="../static/css/autocomplete.css">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

<script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>

</head>

<body bgcolor="#E6E6FA">

<div class="container">

<div class="jumbotron" style="text-align:center">

<h1>Respiratory Diagnose System </h1>

</div>

<div class="row">

<div class ="col-sm-12 col-centered" style="padding-bottom: 1em">

<center>

<div class="btn-group center">

<button id="auto\_btn" type="button" class="btn btn-primary" onclick="getData()">Enter your details </button>

</div>

</center>

</div>

</div>

<form action="/crop\_pre" method="POST" autocomplete="off">

<form action="/crop\_pre" method="post" enctype="multipart/form-data">

<div class="row" style="padding-bottom: 0.5em , margin-left -30px" id="input-value">

<div class = "col-md-3">

<div class="input-group">

<span class="input-group-addon">Age</i></span>

<input type="text" class="form-control" name="Age" placeholder="Age">

</div>

</div>

<div class = "col-md-3">

<div class="input-group">

<span class="input-group-addon">Gender</i></span>

<input type="text" class="form-control" name="Gender" placeholder="Gender">

</div>

</div>

<div class ="col-md-3">

<div class="input-group">

<span class="input-group-addon">BMI</i></span>

<input type="text" class="form-control" name="BMI" placeholder="BMI">

</div>

</div>

<div class ="col-md-3">

<div class="input-group">

<span class="input-group-addon">file</i></span>

<input type="text" class="form-control" name="file" placeholder="file">

</div>

</div>

<br><br><br>

<input type = "file" name = "file" />

<div class="col-md-12" class="">

<br>

<input type="submit" class="btn" value="Diagnose">

</div>

</form>

<div class="col-md-6">

<h3>Disease Prediction </h3>

{% if display == True %}

<div class="table-responsive">

<table class="table table-hover">

<tbody>

<h1 style="color:red">{{predict}}</h1>

</tbody>

</table>

</div>

{% endif%}

</div>

</div>

<script src="../static/js/autocomplete1.js"></script>

<script type="text/javascript">

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

city.style.display = "block"

npk.style.display = "none"

function getData(){

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

// console.log(te.value);

npk.style.display = "block";

city.style.display = "none";

}

function citywise(){

var npk = document.getElementById("input-value");

var city = document.getElementById("city");

// console.log(te.value);

npk.style.display = "none";

city.style.display = "block";

}

</script>

</body>

</html>

**SAMPLE CSS CODE**

#div1{

width:100%;

height:100px;

background-color:#8fcfd1;

border-bottom: 2px solid green;

}

.p1

{

font-size:30px;

color:black;

padding-top:25px;

padding-left:10px;

font-family:"HERCULANUM";

}

**REFERENCES**

1. Welsby, P.D., Parry, G. and Smith, D. ‘The Stethoscope: Some Preliminary Investigations’. Postgraduate Medical Journal (2003)

2. Swarup, S. and Makaryus, A.N. ‘Digital Stethoscope: Technology Update’.

Medical Devices (2018)

3. Sandra Reichert, Gass Raymond, Christian Brandt, Emmanuel Andres ‘Analysis of Respiratory Sounds: State of the Art’. Clinical Medicine. Circulatory, Respiratory and Pulmonary Medicine, (2008)

4. [H Kiyokawa](https://pubmed.ncbi.nlm.nih.gov/?term=Kiyokawa+H&cauthor_id=11399719), [M Greenberg](https://pubmed.ncbi.nlm.nih.gov/?term=Greenberg+M&cauthor_id=11399719), [K Shirota](https://pubmed.ncbi.nlm.nih.gov/?term=Shirota+K&cauthor_id=11399719), [H Pasterkamp](https://pubmed.ncbi.nlm.nih.gov/?term=Pasterkamp+H&cauthor_id=11399719) ‘Auditory detection of simulated crackles in breath sounds’ Chest. 2001 Jun;119(6):1886-92.doi: 10.1378/chest.119.6.1886.(2001)

5. Zimmerman, B. and Williams, D. ‘Lung Sounds’. In StatPearls. Treasure Island (FL): StatPearls Publishing. Available at: <http://www.ncbi.nlm.nih.gov/books/NBK537253/>. (2020a)

6. Zimmerman, B. and Williams, D ‘Lung Sounds’. In StatPearls. Treasure Island (FL): StatPearls Publishing. Available at: <http://www.ncbi.nlm.nih.gov/books/NBK537253/> (2020b)

7. Emmanuel Andres, Amir Hajjam. ‘Advances and Perspectives in the Field of Auscultation, with a Special Focus on the Contribution of New Intelligent Communicating Stethoscope Systems in Clinical Practice, in Teaching and Telemedicine’.EHealth and Remote Monitoring. InTech. (2012)

8. B. M. Rocha, D. Filos, L. Mendes, I. Vogiatzis, E. Perantoni, E. Kaimakamis, P. Natsiavas, A. Oliveira, C. Jácome, A. Marques, R. P. Paiva, I. Chouvarda, P. Carvalho, N. MaglaverasΑ. ‘Respiratory Sound Database for the Development of Automated Classification’ Precision Medicine Powered by PHealth and Connected Health. IFMBE Proceedings. Singapore: Springer Singapore(2018)

9.Renars Xaviero Adhi Pramono, Stuart bowyer, Esther Rodriguez-Villegas ‘Automatic Adventitious Respiratory Sound Analysis: A Systematic Review’ (2017)

10. Singh, A., Thakur, N. and Sharma, A. ‘A Review of Supervised Machine Learning Algorithms’3rd International Conference on Computing for Sustainable Global Development (INDIACom) (2016)

11. Georgios Ntritsos, Jacob Franek, Lazaros Belbasis, Maria A Christou, Georgios Markozannes, Pablo Altman, Robert Fogel, Tobias Sayre,Evangelia E Ntzani, Evangelos Evangelou. ‘Gender-specific estimates of COPD prevalence: a systematic review and meta-analysis’ International Journal of Chronic Obstructive Pulmonary Disease (2018).

**IMAGE REFERENCES**

1.https://www.pngitem.com/middle/ooRiRm\_users-man-woman-man-woman-transparent-hd-png/

2.https://www.pngkey.com/detail/u2w7w7u2t4a9y3r5\_register-user-register-online-icon-png/

3.https://www.pngitem.com/middle/ixmxTbR\_user-login-icon-png-transparent-background-login-icon/

4.<https://www.iconfinder.com/icons/649116/account_details_document_profile_send_upload_user_icon>

5.https://www.pngitem.com/middle/JxwRwJ\_machine-learning-model-png-transparent-png/

6. <https://dryicons.com/icon/gear-icon-13082>

7.<https://thenounproject.com/term/evaluation/1132369/>

8.<https://www.123rf.com/photo_141068088_stock-vector-human-lung-disease-prediction-icon-as-eps-10-file.html>

9.https://www.iconfinder.com/icons/6802116/computer\_accessory\_display\_screen\_monitor\_output\_device\_screen\_icon